PROYECTO FIN DE CARRERA

ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD CARLOS III DE MADRID

INGENIERÍA INDUSTRIAL
AUTOMÁTICA Y ELECTRÓNICA INDUSTRIAL

“SISTEMA PARA EL RECONOCIMIENTO DE ACTIVIDADES”

AUTOR: Jorge Hernández Viorreta
SUPERVISOR (HUT): Prof. Aarne Halme, Dr. Tech.
INSTRUCTOR (HUT): Panu Harmo, MSc.
SUPERVISOR (UC3M): Prof. Fco. José Rodríguez Urbano

Madrid, Abril 2009

Autor:	**Jorge Hernández Viorreta**
Título:	**Activities recognition system.**
Helsinki University of Technology	
Departamento:	**Automation and Systems Technology**
Chair:	**AS-84 Automation Technology**
Fecha y Lugar de Lectura:	**Espoo, 20th Julio 2008**
Páginas:	**98 páginas, 64 ilustraciones, 12 tablas y 3 apéndices**
Supervisor (HUT):	**Prof. Aarne Halme, Dr.Tech.**
Instructor (HUT):	**Panu Harmo, MSc.**
Supervisor (UC3M):	**Prof. Francisco José Rodríguez Urbano, Dr.Tech.**

Abstract:

En este proyecto se ha desarrollado una nueva versión del sistema de monitorización de actividades. El sistema se llama ASED (Sistema Automático para Ancianos y Discapacitados) y consiste en: una red de sensores simples y detectores, un servidor que toma los datos de la red, una aplicación que analiza los datos, una base de datos que almacena los resultados del análisis y una interfaz gráfica de usuario que hace disponibles los resultados a través de Internet.

El primer objetivo de la versión actual fue hacer al sistema más robusto. Para ello se corrigieron numerosos bugs y se cambiaron algunas partes del software previo, se rediseñó la base de datos, se mejoró la aplicación web y se documentó el código del sistema completo.

El segundo objetivo fue hacer al sistema más inteligente. Éste debía ser capaz de detectar actividades ocurridas en el área controlada. En esta parte del trabajo se desarrollaron algunos algoritmos para el reconocimiento de actividades. Con el fin de de diseñar y testar este algoritmo se crearon algunas herramientas: un simulador de red y un testador de algoritmos.

La red de sensores se rediseñó para probar los algoritmos en un escenario real. Se colocaron detectores de movimiento distribuidos homogéneamente por la sala de test, se diseñó y desarrolló un medidor de intensidad eléctrica compatible con la tecnología Linet y se añadieron a la red una estación meteorológica y un par de detectores de haz fotoeléctrico.

Ahora el sistema es estable y detecta algunas actividades, pero se han identificado algunos problemas: los detectores de movimiento no son la mejor opción para detectar presencia; cada vez que se añade un nuevo dispositivo a la red es necesario testarlo y elaborar un preprocesamiento de la señal particular; finalmente se detectó que el tiempo entre recepciones del servidor era demasiado alto para detectar algunas de las actividades probadas.

Keywords: ***Automatización del hogar, Linet, oBIX, Java, Base de datos***

MySQL, JSP, Servicios Web, Detectores de movimiento IR, diseño de circuitos electrónicos.

Person/unit: Aarne Halme, Panu Harmo

EVALUATION OF THE FINAL PROJECT AS-84.FP15

Mr. Jorge Hernández: Activities recognition system

Jorge Hernández prepared his final project at Automation and Systems Technology Department of Helsinki University of Technology (TKK) during the year 2008. He worked in a project called Piloting of Assistive Automation, the goal of which was to study and develop home automation that helps senior citizens to live independently and safely at home as long as possible. Monitoring of a person's activities can give early warning signals or accute alarms of problems at home. The project leader and the head of Automation Technology Research Group was Professor Aarne Halme. MSc (EE) Panu Harmo managed the project. Several other researchers and students also contributed to the development and research work in the project.

Jorge Hernández'es task was to study and to develop a person's activities recognition system based on motion detectors and other sensors that can be installed in home environments. The practical task was to improve a previously developed activities recognition system. The system utilized an Open Building Information Exchange (OBIX) based server, the Open Facility Management Server, which has been developed at TKK. The programs are implemented in Java.

After analysing the existing system Jorge Hernández improved the software system. Then he setup a sensor network for activities recognition tests. This work involved sensor interfacing, wiring, configuring of a Linet network and computer programming. Finally tests of the system and its activities recognition capabilities were performed. The testing of the system showed that it can be used for monitoring various activies indoors. The testing also showed that some of the components, namely the motion detectors, of the system should be replaced or improved in some way.

In the written report Mr. Hernández first describes the system that existed before his work. Then he continues by describing the changes and new features that he implemented. The application and the technology behind it are thoroughly explained. Tests and results of the work are described in the final chapter. The final project document is well structured and well written with many graphs, pictures and tables.

We propose the Final Project of 15 ECTS credits to be approved with the grade v e r y g o o d 4/(5), ECTS grade B.

16th of October, 2008

Aarne Halme, Professor
Department of Automation and Systems Technology
Helsinki University of Technology

Panu Harmo, MSc, project manager
Department of Automation and Systems Technology
Helsinki University of Technology

TEKNILLINEN KORKEAKOULU Automaatiotekniikan laboratorio
TEKNISKA HÖGSKOLAN PL 5400, 02015 TKK
HELSINKI UNIVERSITY OF TECHNOLOGY Konemiehentie 2, Espoo
TECHNISCHE UNIVERSITÄT HELSINKI puh. (09) 451 3301, fax (09) 451 3308
UNIVERSITE DE TECHNOLOGIE D'HELSINKI http://www.automation.hut.fi

RESUMEN EN ESPAÑOL

1. Introducción

La población de los países desarrollados está envejeciendo, en menos de 50 años los segmentos con mayor proporción de habitantes estarán por encima de los 60 años. Este escenario provoca que tengamos que plantearnos nuestra estructura social y económica. Uno de los problemas que tendremos que manejar y solucionar será el cuidado de los mayores e inválidos, problema complejo pues será un grupo heterogéneo, grande y con unos requerimientos especiales en cada caso. Analizaremos las posibles soluciones que tenemos actualmente:

- Residencias y hospitales: Son una buena opción para gente que necesite cuidados continuos e intensivos, con tratamientos específicos difíciles de aplicar en el hogar. Los problemas de esta solución es que los pacientes han de abandonar su entorno y que es un sistema costoso, difícilmente aplicable al número de personar que necesitan ser cuidadas.
- Enfermeros y cuidadores en el hogar: esta propuesta soluciona el inconveniente de tener que abandonar el entorno pero presenta otros problemas, principalmente que es un sistema muy caro que no es sostenible con la previsible reducción de las pensiones al tener menos proporción de población que cotice. Además hay mucha gente que no quiere vivir con un extraño en casa.
- Sistemas de vigilancia por cámara: este sistema es bastante barato hoy en día, los operadores serían capaces de detectar problemas evidentes a través de monitores y de avisar a la persona que pueda solucionar el problema en cada caso (policía, enfermero, ambulancia, etc.). Pero este sistema tiene varias desventajas: primero, a nadie le apetece tener cámaras vigilándole 24h del día, perdiendo intimidad; segundo, sólo problemas muy evidentes podrán ser detectados, pues un solo operados que vigile a varios pacientes no es capaz de seguir y detectar todos los patrones de comportamiento de cada uno, por ejemplo una persona que va al baño más de lo habitual puede denotar un problema, ese “más de lo habitual” es lo que no sería capaz de detectar este tipo de sistemas.

Una vez analizados estos sistemas podemos intuir que características debe tener una solución para considerarla mejor que las actuales:

- Debe ser barato.
- Debe ser eficaz detectando problemas.
- Debe recoger datos de los sujetos sin molestar el día a día.
- No debe causar una pérdida de privacidad.
- Debe ser fácil de instalar en las viviendas de los sujetos.
- Debe adaptarse a las diferentes viviendas, diferentes necesidades y diferentes patrones de comportamiento de cada persona.

- Debe comunicarse con el exterior eficientemente para avisar a las personas que puedan solucionar el problema.
- Debe ser capaz de ser configurado, monitorizado y supervisado remotamente.

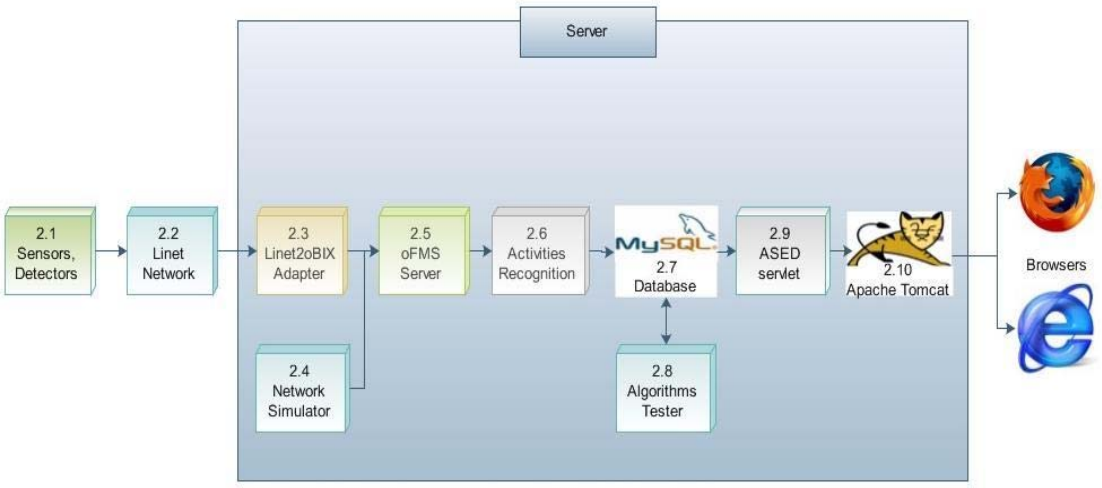
De acuerdo a estos parámetros, se ha desarrollado un sistema llamado Automation System for Elderly and Disabled – ASED (Sistema Automático para Ancianos e Inválidos) cuya segunda versión ha sido el tema de este proyecto. El sistema y sus nuevas características se explican en el capítulo 2 y 3 respectivamente.

2. Sistema Automático para Ancianos e Inválidos. ASEDv2.0

La primera versión de este sistema ha sido descrita en el proyecto de Pablo Ramos. Esta nueva versión ha sido desarrollada con el propósito de hacerlo más robusto y de añadir nuevas funcionalidades.

Este capítulo describe el sistema en su conjunto y el capítulo 3 describe con más profundidad las nuevas especificaciones.

El sistema está dividido en las siguientes partes:



2.1 Sensores, detectores e interruptores: Dependiendo de las actividades que se quieran detectar en cada aplicación se usarán distintos dispositivos. Sensores de movimiento, de humedad, de CO2, de humos, termómetros, etc. habrán de elegirse según queramos controlas las condiciones de “calefacción, ventilación y aire acondicionado” (HVAC), detección de fuegos, seguridad del hogar, control de accesos u otros posibles usos.

2.2 Red Linet: la red Linet es una tecnología propietaria de una compañía local, con sus propios paquetes, su propio protocolo de comunicación, etc. Cada dispositivo de los declarados en el punto 2.1 ha de ser integrados en los nodos Linet. Todos los nodos están conectados al mismo bus al cual también está conectado el controlador Linet. Este controlador se encarga de recolectar

y manejar toda la información de los nodos, también se encarga de conectarse a la red a través de una conexión Ethernet.

2.3 Adaptador para codificar los paquetes Linet en paquetes oBIX (Linet2Obix Adapter): el sistema ASED trabaja con un lenguaje oBIX, que es un estándar. La idea es que el sistema pueda trabajar con cualquier tecnología del mercado (y con varias distintas al mismo tiempo), desarrollando unos programas adaptadores que traduzcan de cada una de esas tecnologías a oBIX y viceversa. En nuestro proyecto hemos usado el "Linet2Obix Adapter", desarrollado por Pablo Ramos en la primera versión de ASED.

2.4 Simulador de red: Con este módulo el desarrollador es capaz de testar el sistema sin una red real. El simulador lee la información acerca de la red que queremos simular a través de un archivo de propiedades XML y muestra una interfaz gráfica donde todos los nodos pueden ser manipulados. Cada tipo de nodo muestra un objeto distinto en la ventana y el estado de todos los nodos se muestra en tiempo real.

2.5 Servidor oFMS: el servidor oFSM fue diseñado por Hannu Järvinen [Jä07]. El servidor recibe los datos del Linet2oBIX Adapter y los guarda en un archivo XML que es accesible vía HTTP. Todo el modelo de petición/respuesta usado viene definido por las especificaciones de oBIX, excepto la función signUp. "Todo método toma una dirección y un string de petición y devuelve un string de respuesta. De ese modo no es necesario preocuparse por el protocolo HTTP ni por ningún otro tema relacionado con la comunicación.

2.6 Aplicación "Activities Recognition": esta aplicación es el software encargado de almacenar la información de los sensores en la base de datos MySQL y de ejecutar los algoritmos para el reconocimiento de actividades (combinaciones simples de eventos detectados en tiempo real) y los algoritmos para la detección de patrones de comportamiento (estadísticos). El resultado de esos algoritmos es también almacenado en el resto de las tablas de la base de datos. La aplicación además contiene un cliente http que hace pregunta al servidor oFSM si ha habido cambios y es capaz de escribir en él.

2.7 Base de datos: se ha usado una base de datos MySQL. En ella se almacenan los datos que le manda la aplicación "Activities recognition". Guarda un log de todos los eventos de los nodos Linet ("nodes_log") y también tiene una serie de tablas donde se almacenan los resultados de los algoritmos.

2.8 Testador de algoritmos: este módulo tiene la utilidad de poder cambiar, analizar y valorar los distintos algoritmos sin interrumpir el funcionamiento del sistema. El testador usa la tabla "nodes_log" para producir los resultados que se obtendrían si se aplicasen distintos algoritmos con las mismas condiciones de la red, con los mismos eventos. Los resultados obtenidos se guardan en tablas alternativas que no interfieren con las tablas usadas en la aplicación "Activities Recognition".

2.9 Aplicación Web: la aplicación web en ASED es básicamente una Interfaz de Usuario Web (WUI), se ha escrito en Java Server Pages (JSP), que combina el código HTTP con líneas en Java. La WUI es bastante importante

para manejar el sistema, pues provee una interfaz amigable donde la información importante de nuestro escenario es accesible y a través de la cual el usuario puede ver los patrones de comportamiento y las actividades de la gente que está dentro de la zona monitorizada.

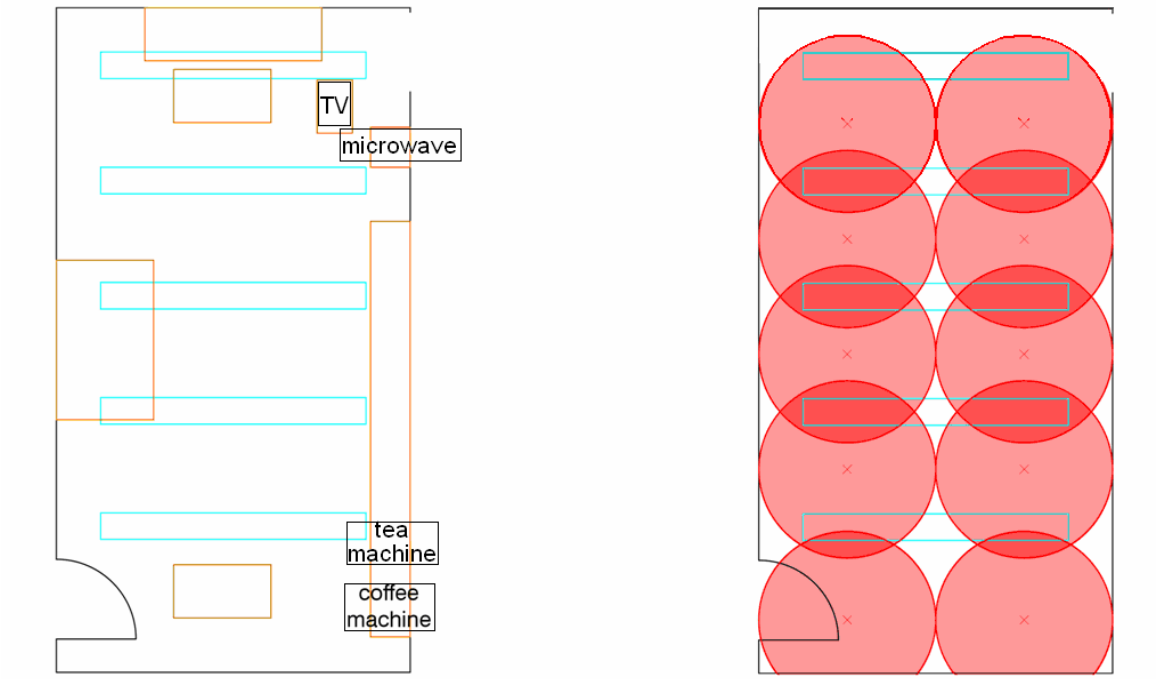
2.10 Servidor Web: se ha usado Apache Tomcat como servidor web pues el proyecto está escrito en Java y Tomcat ofrece un entorno “puro Java” para HTTP. Tomcat implementa Java Servlets y especificaciones JSP.

3. Nuevas características de esta versión

ASEDv2.0 se ha desarrollado con dos propósitos, hacer más robusto el sistema, corrigiendo o modificando numerosos bugs de la versión anterior y añadir nuevas funcionalidades orientadas a la detección de patrones de comportamiento. También se ha diseñado un nuevo escenario para que sea más fácil testear nuevos algoritmos.

En primer lugar se definieron las actividades que es sistema debía reconocer: debería detectar la posición de las personas en una sala, detectar el uso, encendido y apagado de distintos dispositivos eléctricos, también el sistema deberá guardar un recuento de las personas que entren y salgan de la habitación, obteniendo en cada momento el número de personas que hay en el interior. Además sería interesante también analizar las condiciones ambientales de la habitación: temperatura, humedad, CO2, etc.

En segundo lugar se diseñó una nueva configuración de la red de sensores para instalarla en una habitación que usaremos como habitación de pruebas.



Con el fin de detectar el posicionamiento de los sujetos en la habitación se instalaron sensores de movimiento en el techo como muestra la figura adjunta

con el fin de cubrir el máximo espacio con los mínimos solapamientos entre dominios.

Para detectar el uso de los dispositivos eléctricos en la habitación se diseñó y se fabricó un circuito para medir corriente. Estos dispositivos se colocaron entre el conector de alimentación y el enchufe de todos los aparatos eléctricos. El medidor de corriente diseñado estaba compuesto por un circuito rectificador, un filtro RC y un transformador de corriente. El transformador mide la corriente que pasa por el cable neutro o por el de fase, el valor medido por el transformador se rectifica con un puente de diodos y el valor rectificado se filtra. Una vez obtenido un valor en continua, se mandaba al nodo Linet, teniendo en cuenta que ese valor estuviese dentro del rango aceptado por el nodo. Estos medidores de corriente se diseñaron para que pudieran medir distintos rangos de corriente usando dos métodos: cambiando los valores de la resistencia y del condensador del filtro RC ó cambiando el número de vueltas que el cable neutro (o el de fase) da alrededor del toroide del transformador.

Para realizar el conteo de personas se han utilizado tres métodos. Por un lado tenemos dos detectores de haz fotoeléctrico que se colocaron en el marco de la puerta de entrada a la sala, uno en la parte exterior del marco y el otro en la interior, de tal modo que no solo se detecta el paso de alguien a través de la puerta sino que también se consigue detectar la dirección, si tenemos en cuenta que haz se interrumpió primero y cual después. Por otro lado tenemos la capacidad de monitorizar las condiciones ambientales de temperatura y humedad de tal modo que se puede conseguir una relación entre el valor de estos factores y la cantidad de personas en la habitación. Por último también se podían usar la información del resto de dispositivos, pues por ejemplo si el conteo de personas nos muestra que no hay personas en el interior de la “test room” pero la cafetera se enciende, por sentido común podemos decir que por lo menos hay una persona dentro.

A partir de aquí todos los cambios realizados son relativos al software del sistema ASED:

Se configuró la red Linet, asociando cada nodo con el tipo de información que se extraería de él, de ese modo se usaron dos configuraciones básicas: I/O node (nodo que manda un bit de información) y AD/state node (que manda un número entero al sistema).

Se desarrolló un simulador de red. Con esta nueva aplicación es posible testear una red sin tener que montarla físicamente. Los desarrolladores ahora son capaces de comprobar su aplicación cambiando el estado de los sensores y detectores virtualmente, usando una interfaz gráfica. La configuración de la red es leída de un fichero XML de propiedades. La interfaz gráfica muestra todos los nodos en forma de botones y barras de desplazamiento según sean “I/O nodes” ó “AD/State nodes”. El simulador de red conecta directamente con el servidor oFMS, de tal modo que cambiando el estado de un nodo esto hace que cambie el estado de ese nodo en la base de datos interna del servidor.

Se corrigió y mejoró la aplicación para el reconocimiento de actividades “Activities Recognition Web Service” (anteriormente conocida como eCaregiver

Web service). Esta aplicación fue desarrollada y explicada por Pablo Ramos en su Proyecto Fin de Carrera, pero ya que muchas partes han sido modificadas y se han añadido nuevas funcionalidades es necesaria una nueva descripción del sistema.

La aplicación “Activities Recognition Web Service” se estructura en 5 módulos: una Interfaz Gráfica de Usuario (GUI), un módulo de red, un módulo lógico, un módulo de base de datos y un módulo de reconocimiento de actividades.

- Este módulo contiene la función “main”, que es la que inicia la aplicación. Esta interfaz gráfica fue diseñada para mostrar gráficamente todos datos de la red y de los valores obtenidos al ejecutar los algoritmos contenidos en la aplicación, pero en esta nueva versión la interfaz gráfica ha sido movida a la Interfaz Web de Usuario. Ahora este módulo tan solo muestra una ventana preguntando si se desea continuar con la última base de datos usada o s por el contrario se prefiere resetear la misma, borrando todas las tablas y entradas e iniciar la recogida de datos desde ese punto.
- El módulo de red no ha sido modificado respecto a la versión anterior. A través de este módulo se comunica a través de http con el servidor oFMS y se gestionan los “watches” que se aplican a los datos del servidor, de tal modo que solo se envíen los datos cuando el estado del nodo cambia, de ese modo reducimos el tráfico de red.
- El módulo lógico tampoco ha sido modificado respecto a la versión anterior. Este módulo maneja los comandos provenientes del módulo de red y del de reconocimiento de actividades. Esta parte del código lee las propiedades del sistema (dirección del servidor, etc.) a partir de un archivo XML de propiedades y a partir de esa información el resto de módulos pueden acceder correctamente a la información proveniente de la red de sensores y actuadotes.
- El módulo de Reconocimiento de Actividades es el “cerebro” de la aplicación de Reconocimiento de Actividades. Este módulo maneja los paquetes recibidos desde el oFMS, ejecuta los algoritmos de reconocimiento de actividades y almacena los resultados en la base de datos. Se divide en 4 submódulos:
    - DatabaseControler sub-module: establece comunicación con la base de datos MySQL a través del módulo Database y crea las tablas con los valores iniciales adecuados. Esto último solo ocurre si se aceptó la opción de resetear la base de datos.
    - AlgorithController sub-module: se encarga de toda la lógica que hay detrás del submódulo algoritmo. Aquí se realiza el preprocesamiento de los valores recibidos de los sensores, obteniendo valores adecuados para el análisis de las actividades ocurridas. Cada vez que un valor culmina la fase de preprocesamiento correctamente es mandado al submódulo algoritmo y además se almacena una entrada en la base de datos en la tabla “nodes_log”. El preprocesamiento es necesario para filtrar la señal y evitar problemas

de continua variabilidad.10 bits de precisión hace que la más mínima variación de la señal leída en uno de los sensores que reportan un valor analógico provoque una variación del estado de ese nodo en el servidor oFMS. Esto desemboca en un bombardeo excesivo de ese tipo de señales que colapsa la aplicación, la ejecución de los algoritmos y la base de datos. También hay otros problemas como que en ciertos dispositivos al llegar al valor 0 analógico no reportan señal.

  - Algorithm sub-module: implementa los algoritmos para la detección de actividades y para el análisis de patrones de comportamiento. Los resultados se almacenan en las tablas respectivas de la base de datos.

- El módulo de Base de Datos maneja las acciones de aplicación sobre la base de datos MySQL: las consultas, eliminación y creación de tablas y de entradas. Esta es la única clase que usa el driver JDBC. Este driver provee muchas funciones para usar una base de datos MySQL en un entorno Java (Nota: no todas las funciones están implementadas en la versión actual. Por ejemplo se detectó que la función SOURCE, usada para almacenar una base de datos en un archivo externo no funcionaba adecuadamente). Al asumir este módulo todo el código de las consultas, que han sido diseñadas específicamente para esta aplicación, se consigue aligerar mucho el trabajo del sub-módulo Algoritmo.

Por lo que respecta a la aplicación de reconocimiento de actividades solo decir que todo el código desarrollado y el ya existente se ha comentado y documentado, creando una API mediante la herramienta Eclipse.

Otra herramienta desarrollada es el Testador de Algoritmos. Esta herramienta sirve para poder ver los resultados de ejecutar distintos algoritmos sobre la tabla de registros “nodes_log”. Esto permite además poder testar nuevos algoritmos aunque el sistema original esté en funcionamiento y haciendo sus propias actuaciones, las tablas en donde se obtienen los resultados no son las mismas que las que usa es sistema original. También permite testar algoritmos offline y podemos comparar distintos algoritmos teniendo los mismos inputs.

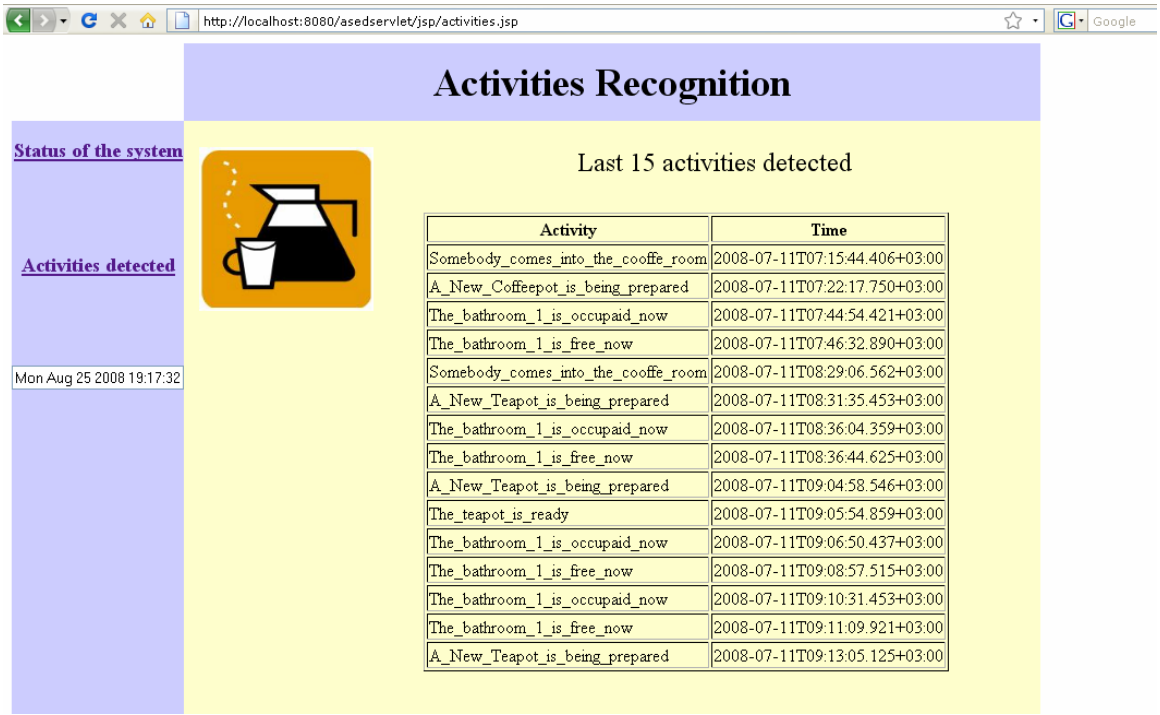
La base de datos también fue rediseñada con el fin de clasificar mejor la información y para facilitar la posterior adición de funcionalidades o la reconfiguración del sistema. Las tablas de la base de datos son:

- Tabla “nodes_log”: esta tabla guarda un registro de todos los cambios de estado significativos de los nodos de la red.
- Tabla “corridor_summary”: el principal objetivo de esta tabla es optimizar las consultas hechas desde la aplicación web para que al mostrar el estado de la habitación test a través de la web tengamos la información en tiempo real. Si no tuviésemos esto habría que hacer consultas complejas sobre una base de datos con muchos registros cada vez que el navegador refrescase la página (2 veces por segundo). Esta tabla solo

es actualizada cuando es necesario y el WUI solo tiene que hacer consultas muy sencillas y rápidas a esta pequeña tabla.

- Tabla “coffeeroom_activities”: en esta tabla se almacena la actividad detectada en la habitación y la hora a la que se produjo.

La última funcionalidad mejorada en el sistema es la Interfaz Web de Usuario, WUI. La WUI es una aplicación web que muestra gráficamente la información a través de un navegador. Las ventajas de tener la interfaz gráfica a través de Internet son la capacidad de acceder a la información desde cualquier ordenador conectado a Internet, la posibilidad de usar la gran veriedad de herramientas de diseño de páginas web, como Adobre dreamweaver, Google Charts, etc. Esta WUI ha sido programada en Java Server Pages (JSPs) ya que necesitamos usar variables que cambian dinámicamente y HTML no soporta esa funcionalidad. Las JSPs se colocan en un servidor de aplicaciones web, en nuestro caso Apache Tomcat Server. Una página JSP es típicamente una página HTML con etiquetas (tags) especiales para incluir código java. La página se compila dando un servlet en segundo plano y funcionando como tal. Un ejemplo de la WUI se muestra a continuación:



4. Test y resultados

Test de los detectores de movimiento: En este test comprobamos como se comportaban estos detectores en función de la distancia y en función de la rapidez de movimiento. Para ello colocamos 4 detectores a 50cm, 1m, 2m y 4m y comprobamos los resultados al pasar por la zona controlada andando despacio, andando rápido, corriendo o estando parados. Además todo este test lo realizamos con 3 tipos de personas que representarían un adulto grande, un

adulto de tamaño normal y un niño. Los resultados mostraron que siempre que hubiese movimiento, el que fuese, los resultados eran bastante satisfactorios en todas las distancias, pero que estando de pie enfrente de los detectores, en el mejor de los casos (adulto de talla grande), era de tan solo el 55%. Con todos estos datos también podemos decir que este tipo de detectores no funcionan bien para poder calcular la distancia a la que se produjo el movimiento pues los resultados en todas las distancias eran muy parejas.

Test de conteo de personas: en este test se evaluó la efectividad del método de conteo a partir de detectores de haz fotoeléctrico. En este caso se evaluó con una persona entrando y saliendo de la habitación a distintas velocidades. Los resultados mostraron que el método funcionaba mal, y curiosamente funcionaba peor en el caso de entrar en la habitación que al salir de ella. Tras un análisis exhaustivo se determinó que esto era debido a que el tiempo de refresco del sistema era mayor que el tiempo que pasaba entre que se activaba un detector y el otro. Esto provocaba que el sistema recibiera el cambio de valor de ambos nodos al mismo tiempo, y según la programación, cuando esto ocurría colocaba el sensor exterior después del interior. Esto explica por que detectaba mejor las salidas de la habitación que las entradas.

Test de localización de personas: En este test planteábamos una serie de preguntas que condicionarían el desarrollo de los algoritmos de detección de actividades y de la localización de personas en la habitación. Los resultados mostraban que era muy difícil localizar con precisión a personas en la habitación por diferentes motivos: los dominios de muchos detectores se solapaban demasiado unos con otros, en la habitación el movimiento de las personas no era continuo, lo que hacía que los detectores estuviesen cambiando de estado todo el rato. Con la configuración actual y haciendo algún preprocesamiento de la señal quizás se podría localizar vagamente una sola persona.

Test de uso de aparatos electrónicos: en este test se evaluó la efectividad de los medidores de intensidad para medir la actividad o no de aparatos eléctricos. Se hicieron dos test para cada aparato: uno donde se encendía el aparato y se dejaba encendido 5 minutos y luego se apagaba, y otro donde el aparato se encendía y apagaba 10 veces en 2 minutos aproximadamente. Los resultados de el test de las 10 veces fueron en todos los casos correctos y los resultados del test en régimen permanente también fueron correctos salvo en el caso de la TV. El problema con la TV era del preprocesamiento de la señal, pues se detectó que el medidor de intensidad cuando no detectaba corriente no mandaba un paquete al servidor.

Test de estabilidad del sistema: en este test se dejó funcionando el sistema 15 días para ver la estabilidad del sistema y los resultados fueron satisfactorios, no hubo ningún problema ni error. Si acaso se empezaba a notar un cierto ralentizamiento debido al incremento de registros de la tabla “nodes_log” de la base de datos, que hacía que algunas consultas fueran bastante costosas en tiempo de programación. Una buena idea para el futuro sería limitar la cantidad de filas que los algoritmos usan en sus consultas.

5. Conclusiones

Primero vamos a analizar los problemas detectados en la versión actual:

- El tiempo de refresco del sistema es demasiado alto para detectar actividades muy rápidas.
- Para muchas actividades donde haya movimiento, pero no continuado, los detectores de movimiento no son la mejor opción.
- Los medidores de corriente no envían su estado al sistema cuando este es 0.

Seguidamente se proponen pasos a realizar para el futuro del sistema:

- Con las herramientas añadidas al sistema y solucionando los problemas detectados el siguiente paso sería desarrollar algoritmos más complejos para detectar patrones de comportamiento. Quizás en vez de intentar ser muy concretos en las actividades que se desean detectar, se podría avanzar por el análisis estadístico del estado de los sensores de movimiento, por ejemplo, calculando cuotas de cantidad de registros de movimiento detectados por los detectores en las distintas zonas y en las distintas horas del día. De ese modo se podría ver la variación de esas cuotas y ver para que valores de ellas significa la existencia de un problema en el anciano o inválido
- Se podrían usar técnicas de minería de datos para encontrar los patrones del sistema, que aunque no evidentes, caracterizan al sistema y la variación de los cuales podría denotar un problema en el individuo monitorizado

HELSINKI UNIVERSITY OF TECHNOLOGY
Automation Technology Laboratory

Jorge Hernández

Activities recognition system

Helsinki University of Technology
Automation and Systems Technology
Supervisor at the HUT: **Prof. Aarne Halme, Dr. Tech.**
Instructor at the HUT: **Panu Harmo, M. Sc.**
Supervisor at the UC3M: **Prof. Francisco José Rodríguez Urbano, Dr.Tech.**

UNIVERSIDAD CARLOS III DE MADRID

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Jorge Hernández Viorreta
Title of Thesis:	Activities recognition system.
Helsinki University of Technology	
Department:	Automation and Systems Technology
Chair:	AS-84 Automation Technology
Date and Place:	Espoo, 28th July 2008
Pages:	98 pages, 64 illustrations, 12 tables and 3 appendix
Supervisor (HUT):	Prof. Aarne Halme, Dr.Tech.
Instructor (HUT):	Panu Harmo, MSc.
Supervisor (UC3M):	Prof. Francisco José Rodríguez Urbano, Dr.Tech.

Abstract:

In this Thesis a new version of an activity monitoring system has been developed. The system is called ASED (Automation System for Elderly and Disabled) and consists of: a network of simple sensors and detectors, a server that gets the data from the network, an application that analyzes the data, a database that stores all the results of the data analysis and a graphical web application that makes available the results through internet.

The first goal in the current version was to make the system more robust by correcting some bugs and changing some pieces of the previous software system, redesigning the database, improving the web application and documenting the whole code.

The second goal was to make it smarter. The system should be capable of detecting activities occurring in the controlled area. In this part of the work an algorithm for activities recognition has been developed. In order to design and test this algorithm there were created some tools: a network simulator and an algorithms tester.

The network was redesigned for testing the algorithms in a real scenario. Motion detectors were placed in proper positions in the test room, a Linet-compatible current meter was developed and a meteorological station and a pair of beam photoelectric detectors were added to the network.

Now the system is stable and detects activities, but the some problems of the system have been identified. The motion detectors are not the best option for detecting presence. Everytime a new device is added to the network it is necessary to test it and correct its mistakes in the preprocessing step. Finally the time between receptions from the server is too long for detecting some activities.

Keywords: *home automation, Linet, oBIX, Java, MySQL Database, JSP, Web Services, IR motion sensors, circuit board design.*

Acknowledgements

First I want to thank the Erasmus program, the Universidad Carlos III de Madrid and the Helsinki University of Technology for letting me to live this amazing experience. Learning a new culture and the contact with people from all around the world have enriched me even more than I expected. Finland, a piece of you will always be in me.

I am also grateful to the staff of the Automation Department, who received me very friendly, helped me when I needed and with whom I have worked very comfortably these months. Panu Harmo, Soile Saloranta, José Vallet, Eric Halbach, Mikko Elomaa, Antti Liesjärvi, Jan Philipp... Thanks!

Finally I give thanks to my family and to my girlfriend because, even in the distance, I always felt their affection and support.

Contents

Abbreviations

ASED	Automation System for Elderly and Disabled
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HVAC	Heating, Ventilation and Air-Conditioning
J2EE	Java 2 Enterprise Edition
JSP	Java Server Pages
JVM	Java Virtual Machine
LAN	Local area network
oBIX	open Building Information eXchange
oFMS	open Facility Management Server
OS	Operative Systems
RDBMS	Relational Database Management System
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol Internet Protocol
UDP	User Datagram Protocol
WS	Web Services
WUI	Web User Interface
XML	eXtensible Markup Language

List of Figures

List of Tables

1 Introduction

The population of the "first world" is aging, in the close future a high percentage of the society will be elderly. As seen in the population pyramids of the Figures 1.1, 1.2 and 1.3 in less than 50 years the age groups with most population are over 60 years old. The actual social system maybe will not work in the future, probably some changes in the pension structure and some adjustements in the economy for this new scenario will be necessary.

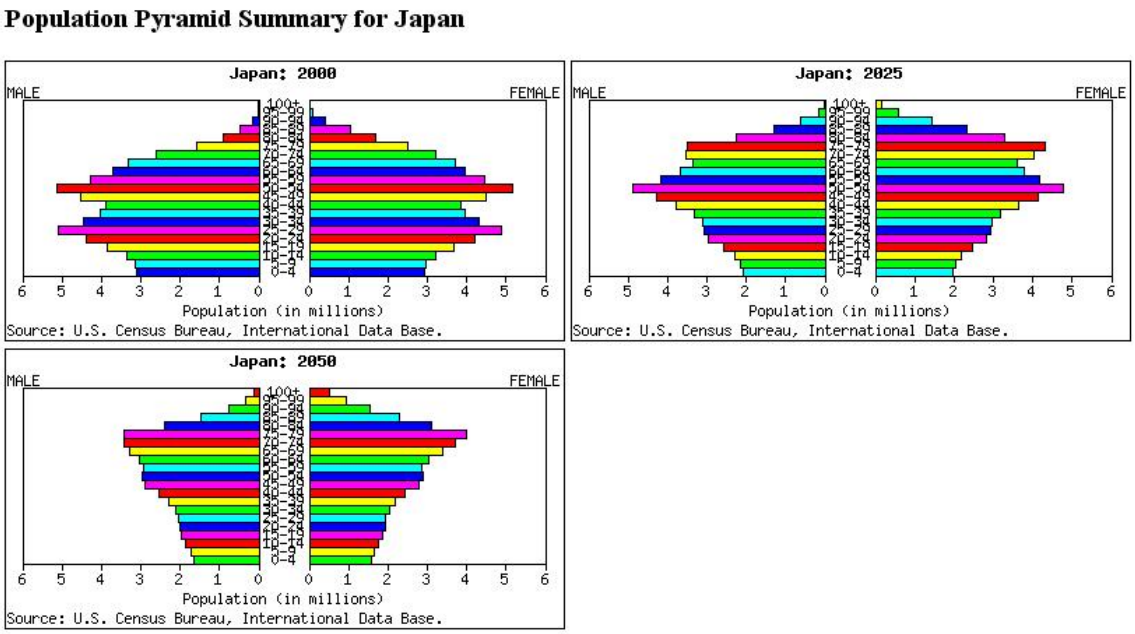


Figure 1.1: Population pyramids of Japan

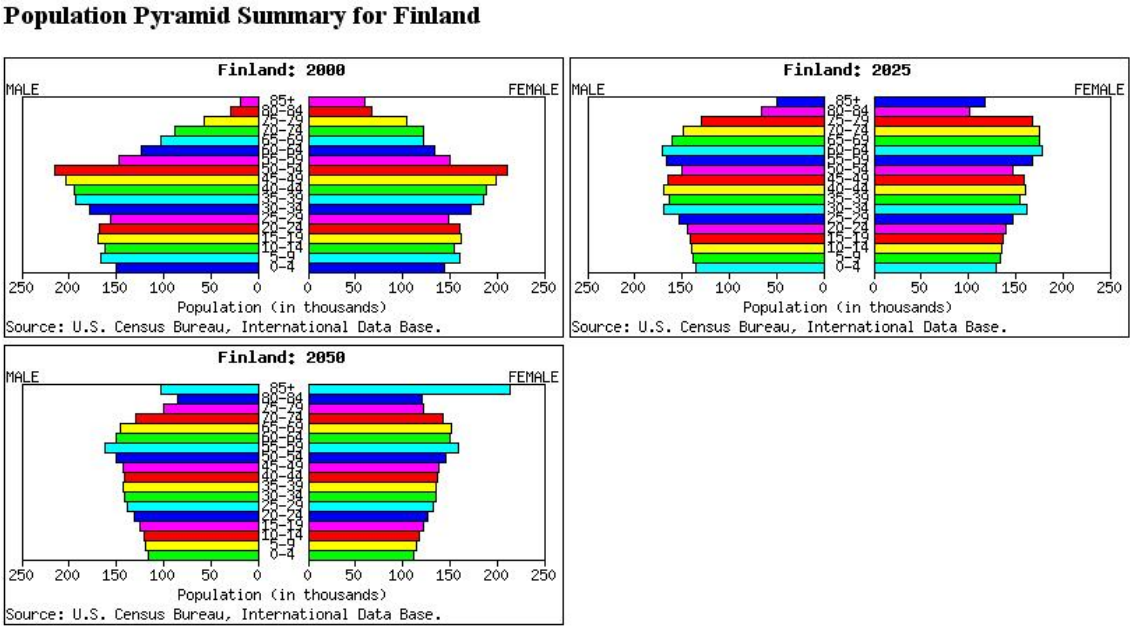


Figure 1.2: Population pyramids of Finland

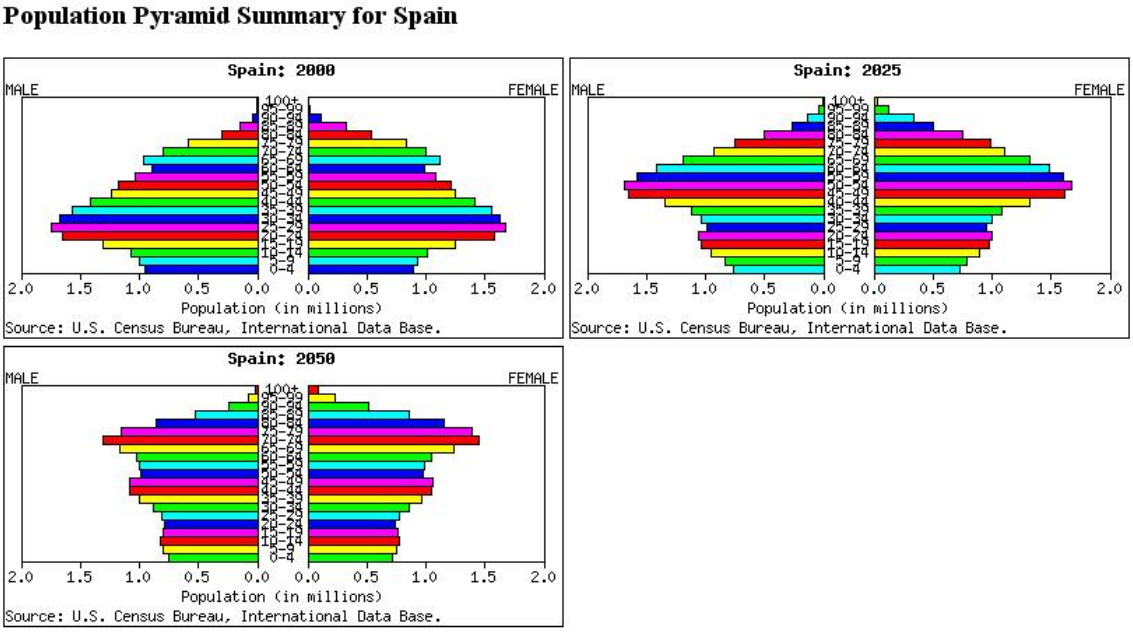


Figure 1.3: Population pyramids of Spain

One of the problems the society has to manage and solve is the caring of this big group of people, even bigger if the disabled are included. They are a heterogeneous group with different needs, different motivations, different ways of life, different familiar structures, different levels of wealth, etc.

Nowadays it is possible to find some solutions for solving the caring of the elderly and disabled. These solutions are analyzed here in order to find their advantages and drawbacks:

- Nursing homes and hospitals: These places are a very good option for people who need continuous care or who require specific treatments that are neither easy nor convenient to do them at home. Its main drawback is that the people should leave their homes, friends in the neighborhood, their environment. Most of elderly and disabled people don't need this special care and they would prefer to be in their own homes.

- Caregivers and nurses at home: This option allows the elderly and disabled people possibility to live in their homes. This option is not sustainable for all the society because the fact of having one person caring is a quite expensive solution and ,as was already explained, the population is getting older so probably the elderly will receive less pensions because there are less people in working ages to pay the taxes. Furthermore, there are a lot of people who do not want to live together with a stranger.

- Camera surveillance systems: These systems are a cheap solution for caring people. The cameras are one resource quite used these days and it is easy to build a system of Close Circuit Television (CCTV). In this system the caregivers can observe the activity of the persons in houses at the same time and they can alarm a nurse, the police, the relatives or

the people who can help when it is necessary. But this system has many drawbacks. First, nobody wants to have cameras watching them all the time, in order to make the system effective in detecting problems, the cameras should cover the whole dwelling, which implies a big loss of privacy. Second, with this kind of systems only the obvious problems can be detected. It is nearly impossible that the people that are watching the monitors are able to follow all the behavioral patterns of all the observed people and detect when something is not normal. For example if somebody is going many times to the bathroom, more than usual, probably he/she is ill or has a problem, this "more than usual" is what the observers are not capable to detect.

After the analysis of all these solutions with their deficiencies and strengths it is possible to determine the aspects that should be achieved to provide a better alternative. The requirements are:

- It should be inexpensive.
- It should be effective in detecting problems.
- It should collect data from the subjects without disturbing their daily life.
- It should not cause a loss of privacy.
- It should be possible to install it in peoples homes.
- It should be adaptable for different houses and different people with different behavioral patterns.
- It should be communicate with the outside world to warn efficiently the people who are able to solve the problem of the subject.
- It should be capable to be configure, monitored and supervised remotely.

According to these parameters, a system is being developed at the Helsinki University of Technology, TKK. This system is called Automation System for Elderly and Disabled (ASED) and its second version has been the subject of this thesis. The system and its new features are explained respectively in the chapters 2 and 3.

2 Automation System for Elderly and Disabled (ASEDv2.0)

The Automation System for Elderly and Disabled is a home assistance system developed at the Automation Technology Laboratory of TKK. The first version of this system has been described in the thesis of Pablo Ramos [Ra07]. This new version was developed in order to make the system more robust and for giving it new functionalities (these new functionalities are detailed in the Chapter 3).

The setup of the whole system consists of the following parts that are explained in this chapter.

(2.1) Sensors, detectors and switches

(2.2) Linet Network

(2.3) Adapter for translating Linet packages into oBIX packages (Linet2Obix Adapter).

(2.4) Network simulator

(2.5) Home server (oFMS)

(2.6) Activities Recognition WS application

(2.7) MySQL Database

(2.8) Algorithms tester

(2.9) Web Application (ASED servlet)

(2.10) Web Server (Apache Tomcat)

A schematic of the system setup is shown in Figure 2.1.

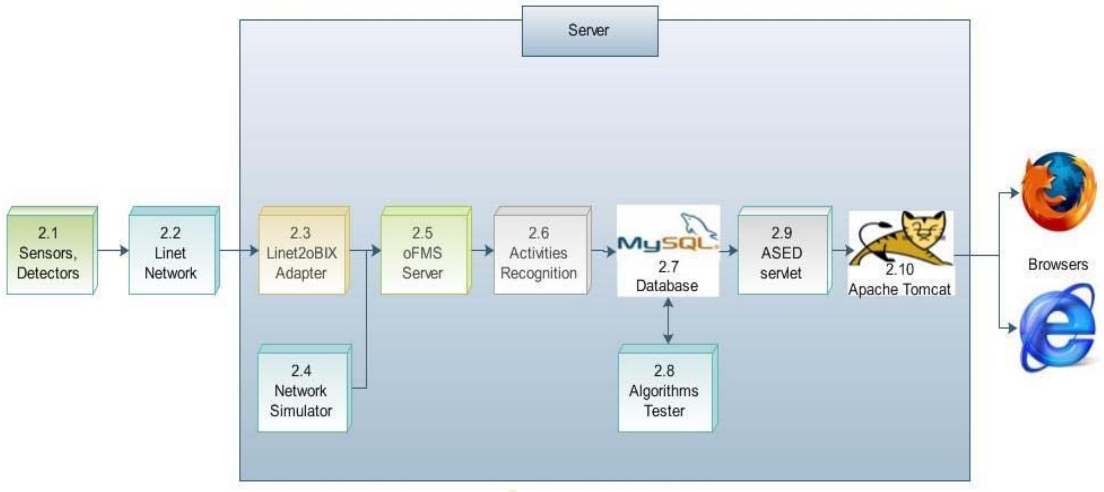


Figure 2.1: General system structure

2.1 Sensors, Detectors and Switches

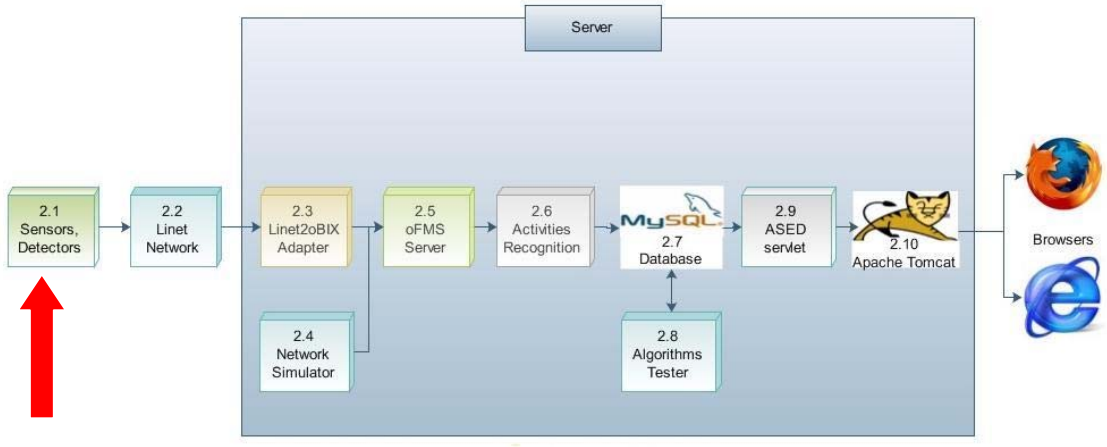


Figure 2.2: Sensors and detectors within the general system structure

In each implementation, depending on what activities or variables will be analyzed, different types quantities and qualities of devices should be chosen.

Here is a list of possible devices, their performances and some possible uses:

2.1.1 Motion Detection

2.1.1.1 Sensor types and features

- IR Motion Detectors: These units detect objects that generate heat in the infrared spectrum (such people and animals) within the detection range.
- Dual Technology Motion Detectors: These Motion Detectors utilize dual technology sensors combining Stereo Doppler Microwave Technology with a dual element Passive Infrared.
- Outdoor Motion Detectors: Motion detector designed for remote location mounted outdoor. Minimize false alarms with sunlight-immune outdoor motion detectors.
- Radar Motion Detectors: Range-controlled radar motion detectors respond to motion only within an adjustable range, and they ignore incidental motion outside that range.
- Wireless Motion Detectors. These devices can work without wiring and have battery life up to 5 years.
- Pet Immune Motion Detectors. These units detect objects that generate heat with infrared radiation, such as people within the detection range, and are immune to animals.

2.1.1.2 Possible uses

- If the user of the ASED system has pets the system probably will need a Pet Immune Motion Detector.
- If the system needs more effectiveness because there are critical places (stairs, shower, etc.), it should use Dual Technology Motion Detector or Radar Motion Detectors.
- The wireless detectors are useful for difficult-to-access places.

2.1.2 Heating, Ventilation and Air Conditioning (HVAC)

2.1.2.1 Sensor types and features

- Water Temperature Sensor
- Room temperature Sensor
- Humidity Sensor
- Water Alarms & Flood Sensors
- CO2 Detectors
- CO Detectors
- Flammable Gas Detectors

2.1.2.2 Possible uses

- The management of the protection against flammable gas leaks or bad heating combustion (resulting in Carbon Monoxide-CO). The system could turn on the fans of the ventilation system, open the windows and cut the electricity to avoid an explosion for flammable gases.
- Improvement of the management of comfort in terms of energy saving, measuring the temperature and humidity in the places where it detects motion activity and controlling the ambient conditions of a single room.
- Notify the user in case of flood and stop the water supply.

2.1.3 Fire Detection

2.1.3.1 Sensor types and features

- Smoke Detector: Photoelectric: measuring the refraction of the light.
- Flame Detector
    - Measuring the radiation frequency.

    - Measuring the energy of the flames.

- Heat Detectors:
    - Fixed temperature (over this temperature it triggers).
    - Speed at which temperature increases.

2.1.3.2 Possible uses

- The system could notify the user and the fire station if it detects a fire and provide some information, if there are people inside the house and in which room they are. It also could activate an anti-fire system, if it exists.

2.1.4 Security

2.1.4.1 Sensor types and features

- Contacts Doors and Windows: Each sensor features a magnetic seal that detects the opening/closing of a door or window.

- Glass Break Detectors

- Photoelectric Detectors: Element that allows us to warn about the passage of an object across a light beam. It senses when the beam is interrupted by the subject.

2.1.4.2 Possible uses

- ASED could notify the user and the police if it detects a broken window.

- There could be an automatic system of opening and closing doors and windows addressed to people with low mobility. It could be checked by motion or presence sensors, supported by contact sensors and be accessible by an interface.

- Photoelectric detectors could be used for counting people or to detect intruders.

2.1.5 Access Control

2.1.5.1 Sensor types and features

- Biometric Readers

- Fingerprint Readers

- Hand Readers

- RFID (Radio Frequency Identification) Readers and Cards

2.1.5.2 Possible uses

- Some people with disabilities are unable to open/close doors with traditional keys. These devices could make this action easier. They would not need to open the door manually, but just to activate the access control. Additionally you would not need to search for the right key any more.

- In case of losing an RFID card, the user could disable this card in the database using his mobile phone (in real time). The customer does not need to change all the locks nor all the keys for these locks.

- Installation of an automatic access control to private areas. The biometric parameters are different for everyone so that these systems are more secure

2.1.6 Close Circuit Television (CCTV)

2.1.6.1 Sensor types and features

- Cameras

2.1.6.2 Possible uses

- These systems can be useful for the management of the people who call at the main door. The user could see who is ringing and open the door or keep it closed. It could be done from anywhere if the customer has a phone connected to ASED.

- It could provide information in case of emergency.

2.1.7 Energy Saving

2.1.7.1 Sensor types and features

- Light Detector

- Rain Sensor

- Electric Power Measuring Devices

- Ultrasound Distance Sensor

2.1.7.2 Possible uses

- The system will save money and energy if it turns off the outside lights during the day and if it does not water the garden during raining days.

- The system can check if a bulb is broken if it is turned on and the light detector does not detect anything or if the current meter does not measure any current value.

- The user could check the consumption of electricity on his mobile phone. This information could be used to find what activities consume more energy and try to reduce them.

- This device could be used to manage the level of a water tank or a cistern.

2.2 Linet Network

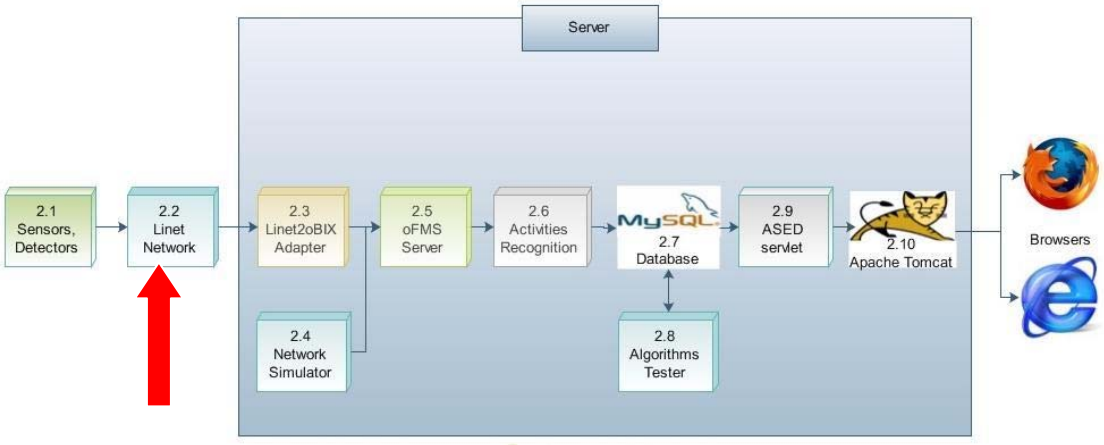


Figure 2.3: Linet Network within the general system structure

The Linet network is a proprietary technology by the Linet company. The network has its own communication protocol, own packets structure, own timers, etc.

Each device has to be connected or integrated in a Linet node and all the nodes are connected to the same bus. The Linet controller manages the collection of all the data that the nodes get from the sensors, detectors, switches, etc. The bus consists of a twisted pair that carries the signal and the power for the nodes.

Sometimes it will be necessary to use an external power supply for the sensors/detectors, but not for the Linet nodes.

If the sensor/detector provides an inappropriate signal for the standard nodes, it could be necessary to use a relay. Linet also provides a node with a solid state relay integrated.

The controller has an Ethernet connection and it implements the TCP/IP protocol so that the network is accessible through a Local Area Network (LAN) or through the Internet.

A full description of the Linet technology is explained in the Appendix B.

2.3 oBIX Adapter

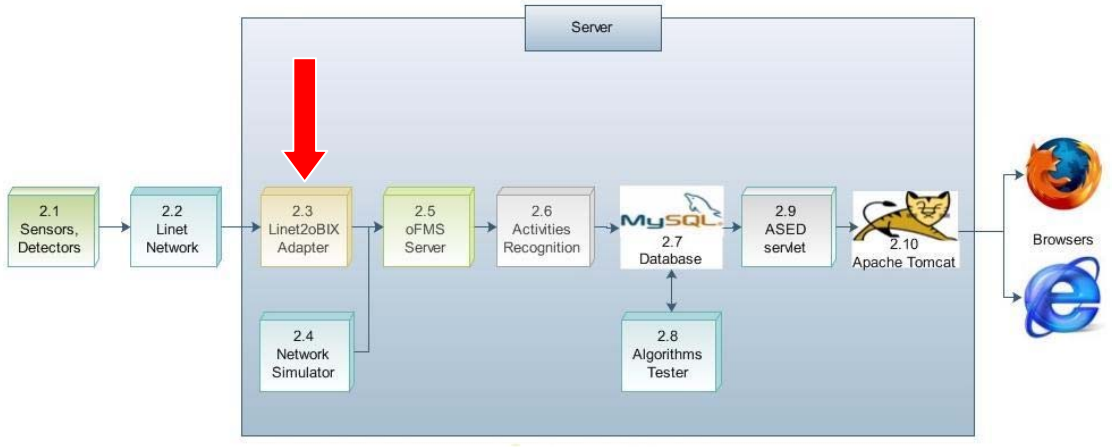


Figure 2.4: Linet2oBIX Adapter within the general system structure

Linet is a proprietary technology, and one of the characteristics of ASED systems is to be able to work with many different technologies; that is why oBIX is the language the application and the server uses. oBIX is a language based on XML and oriented towards building automation. oBIX provides a convenient way to define objects and abstractions and enables the representation of the information from different building automation systems in a standardized way. A bigger explanation of this technology can be found in the Apendix C.

The oBIX adapter used is an adapter for Linet-oBIX. It was designed by Pablo Ramos [Ra07] and it is a piece of software between the Linet controller and the Home Server. It is capable of translating from one language to the other (in both ways) and creating the correct packages on each side.

With these kinds of adapters the system is able to manage different networks and different network technologies at the same time with the same home server. Now Adapters have to be developed for each network technology.

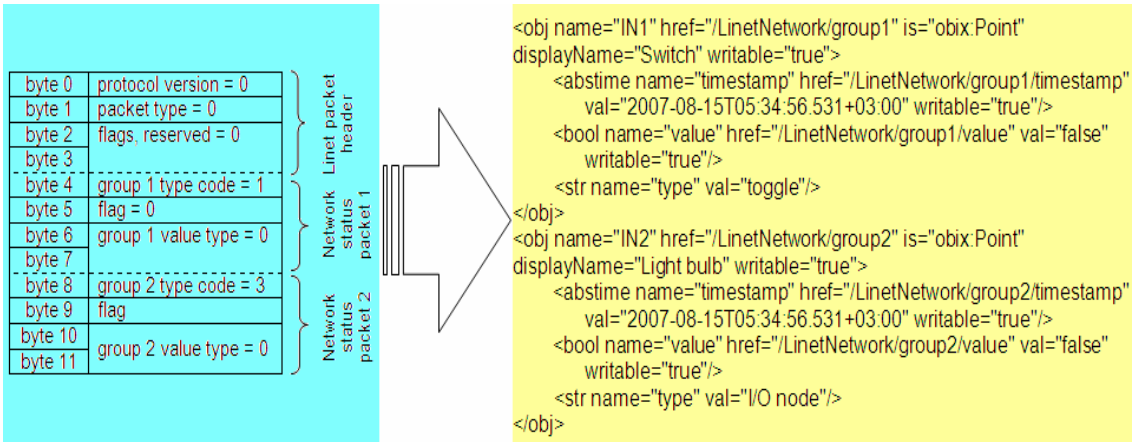


Figure 2.5: Mapping Linet data to oBIX data [Ra07]

2.4 Network Simulator

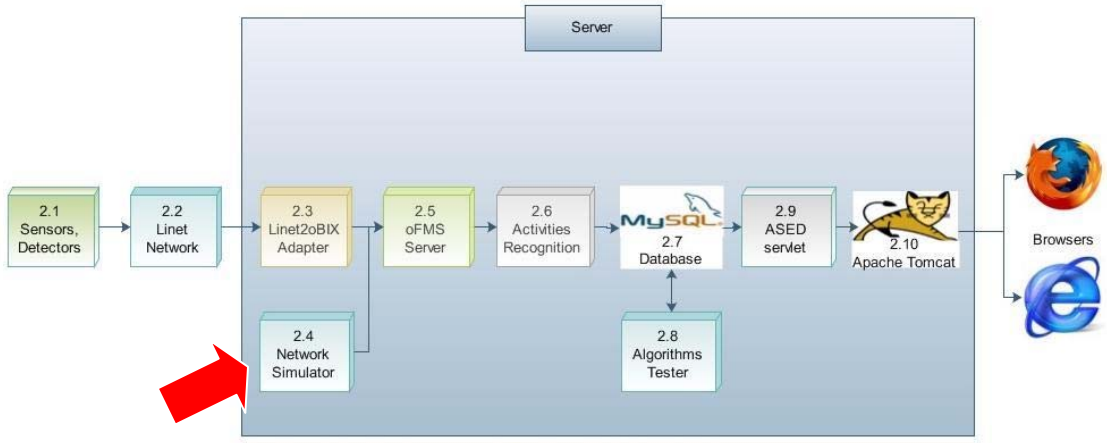


Figure 2.6: Network Simulator inside the general system structure

With this module the developer is able to test the system without a real network. The simulator reads the information of the simulated network from a XML properties file and draws a graphical interface where all the nodes can be manipulated. Each type of node (type of group) has a different graphical object in the window and state is being showed in all of the nodes in real time.

When any node is manipulated the information of the new state of the node is sent to the oFMS server like a real network does it.

Besides, a HTTP client is integrated into this tool so that the simulated network can be operated remotely.

The internal operation and the functionalities will be explained in the next chapter (Section 3.5).

2.5 Home Server (open Facility Management Server - oFMS)

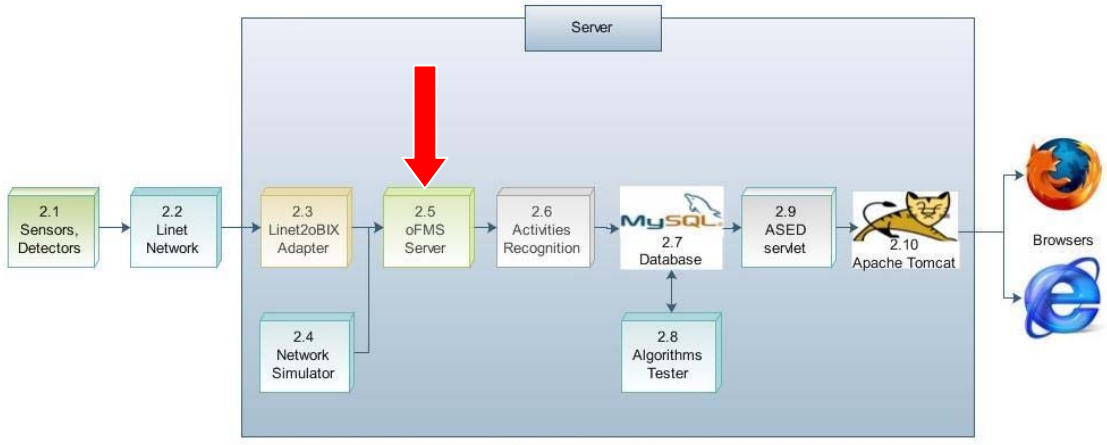


Figure 2.7: oFMS within the general system structure

The Home Server used was designed by Hannu Järvinen and it is completely explained in his Thesis [Jä07], here it is explained briefly to understand the ASED system.

The oFMS receives the data from the Linet2oBIX Adapter, and stores it in a XML file. The data is available via HTTP for the Home Services.

This server has a database for storing the data received from the networks and managed by the services.

The request/response model used to communicate with the oFMS is defined in the oBIX specification, except the signUp function.

“Every method takes an address and a request string as arguments and returns the response string. Thus there is no need to worry about the HTTP protocol or other communication related subjects”. [Jä07]

2.6 Activities Recognition WS

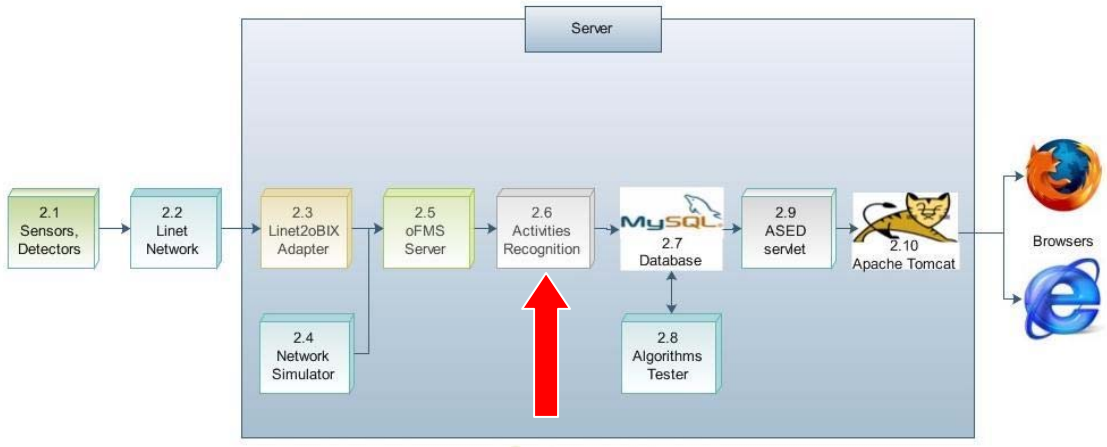


Figure 2.8: Activities Recognition WS within the general system structure

This application is the software in charge of storing the information of the sensors in the MySQL database and of running the algorithms for the recognition of the activities (simple combinations of events detected in real time) and for the detection of behavioral patterns (made from statistics). The result of these algorithms is also stored in the rest of the tables of the database. The application also contains an HTTP Client which polls the oFSM server for changes and is able to write to it.

The internal operation and the functionalities will be explained in the next chapter (Sections 3.6 and 3.7).

2.7 MySQL Database

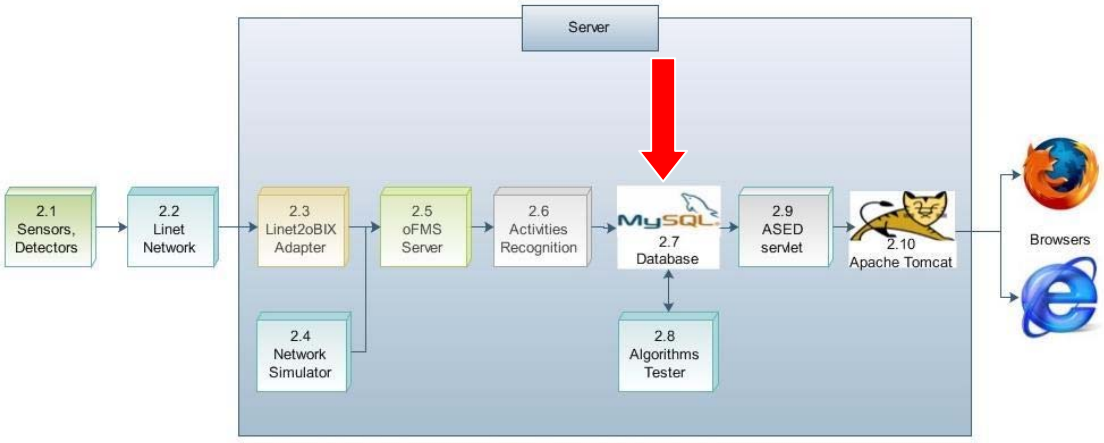


Figure 2.9: Database inside the general system structure

MySQL is a relational database management system (RDBMS). The project's source code is available under terms of the GNU General Public License, as well as under a variety of proprietary agreements. This database is available for almost all the versions of Windows, for Linux and for many other operating systems.

The ASED MySQL database contains some tables with information of the activities that occurred in the monitored area. There is one table, “nodes_log”, where all the information sent by the network is stored. The rest of the tables contain the results of the running algorithms for recognizing activities and behavioral patterns.

The design of the database is explained in the next chapter (Section 3.9).

2.8 Algoritms tester

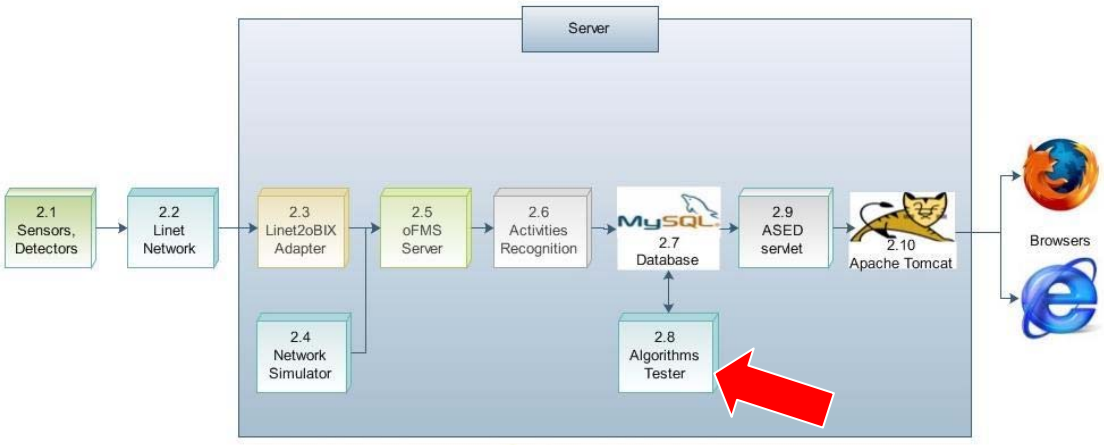


Figure 2.10: Algorithms Tester within the general system structure

This application is a useful piece of software for designing, improving and changing the algorithms for "activities recognition" and "behavioral patterns recognition".

The tester program reads the table of the database where all the events are stored (“nodes_log”) and fills the rest of the tables according to the algorithm that the application is trying. After the application finishes its process of running the tested algorithms, the developer should analyze the results manually.

The “Algorithms Tester” is able to work even when the rest of the system is already running: it creates alternative tables in the database and it does not disturb the running of the rest of the ASED processes.

The internal operation and the functionalities will be explained in the next chapter (Section 3.8).

2.9 Web Application (ASED Servlet)

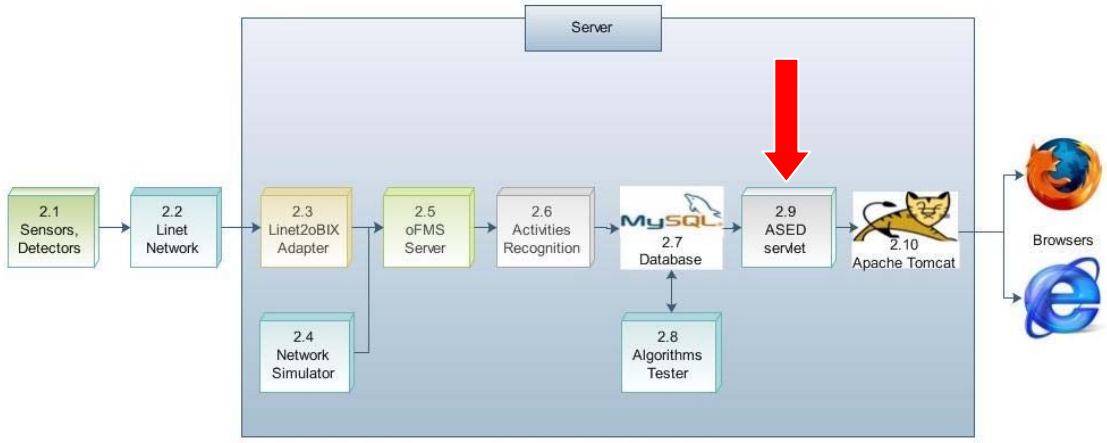


Figure 2.11: ASED servlet within the general system structure

The web application in the system is mainly a Web User Interface (WUI), it is written in Java Server Pages (JSP) which combines the HTTP code with lines in Java.

This piece of software is quite important for the managing of the system, it provides a friendly interface where the important information of our scene is accessible in near-real time, and it is also the place through which the user is able to look for the behavioral patterns and the activities that the people inside the room are doing.

The internal operation and the functionalities will be explained in the chapter (Section 3.10).

2.10 Web Server (Apache Tomcat)

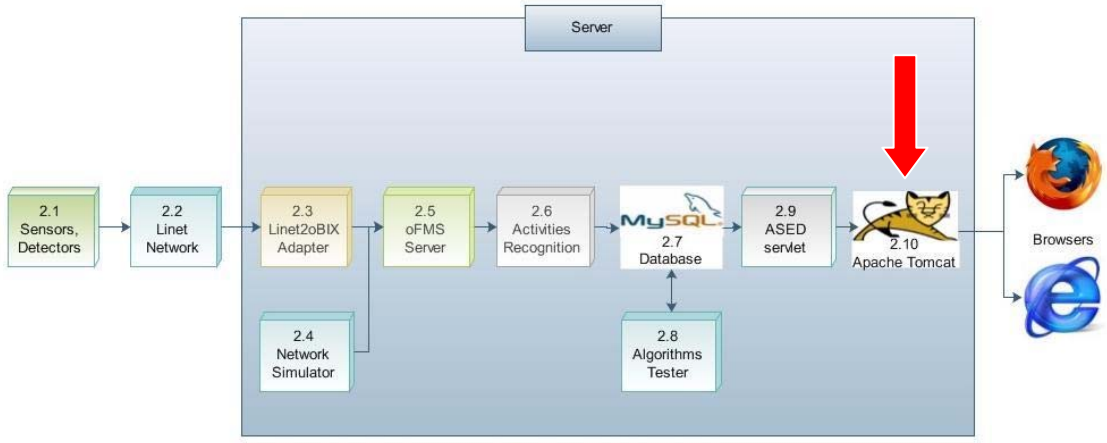


Figure 2.12: Apache Tomcat within the general system structure

The Web server used is Apache Tomcat because the project is written in Java and Tomcat provides a "pure Java" HTTP web server environment for Java code to run. Tomcat implements the Java Servlet and the JSP specifications.

Tomcat should not be confused with the Apache web server, which is a C implementation of a HTTP web server

3 New features

The first version of the ASED system goals were communicating with the system through the Internet (a web application) and collecting, storing, managing and distributing the obtained data through a server.

This new version is oriented towards recognition and detection of behavioral patterns. A new scenario was designed where it was easier to try new algorithms, and tools were developed for managing and controlling the data obtained. The Home Services application was improved and named Activities Recognition WS. Algorithms for activity recognition and detection of behavioral patterns were developed. Finally, the Web User Interface (WUI) was redesigned to provide ASED with a useful tool for viewing the results of the algorithms and for monitoring the data collected by the sensor network.

List of the features developed in this Thesis project:

(3.1) List of the activities the system should recognize

(3.2) Design and implementation of a new sensor network in the coffee room

(3.3) Design and implementation of Linet-compatible current-meters

(3.4) Configuration of the Linet Network

(3.5) Design and implementation of a network Simulator

(3.6) Improvement of the Home Services application to Activities Recognition Web Service (before known as eCaregiver Web Service)

(3.7) Design and implementation of algorithms for activities' recognition and behavioral patterns monitoring

(3.8) Design and implementation of an Algorithms Tester application

(3.9) Design and implementation of the application Database

(3.10) Design and implementation of a new Web User Interface

3.1 Activities the system could recognize

3.1.1 Detecting peoples location

The idea is to put motion detectors covering the whole room and then get the location of the people according to which detector has been activated.

3.1.2 Use of electrical devices

By using current meters the system will be able to know which electric device is being used at each moment.

3.1.3 Counting people in a room

By using 2 photoelectric beam detectors located at a door, one outside the room and the other inside, the system should be able to detect whether someone is entering or coming out from the room.

In order to make the system more robust the system could reset the count to zero when some conditions into the room occur (no motion for a certain period of time) and it could try to count people who are in the room doing measures of the temperature or the humidity.

3.1.4 Changes on the environmental conditions into the room

By using temperature sensors, CO2 sensors, humidity sensors, etc.

3.2 Design of a new sensor network in a test room

Here is an explanation of the types and locations of sensors and detectors used by the network and how the system works towards detecting the activities listed in section 3.1.

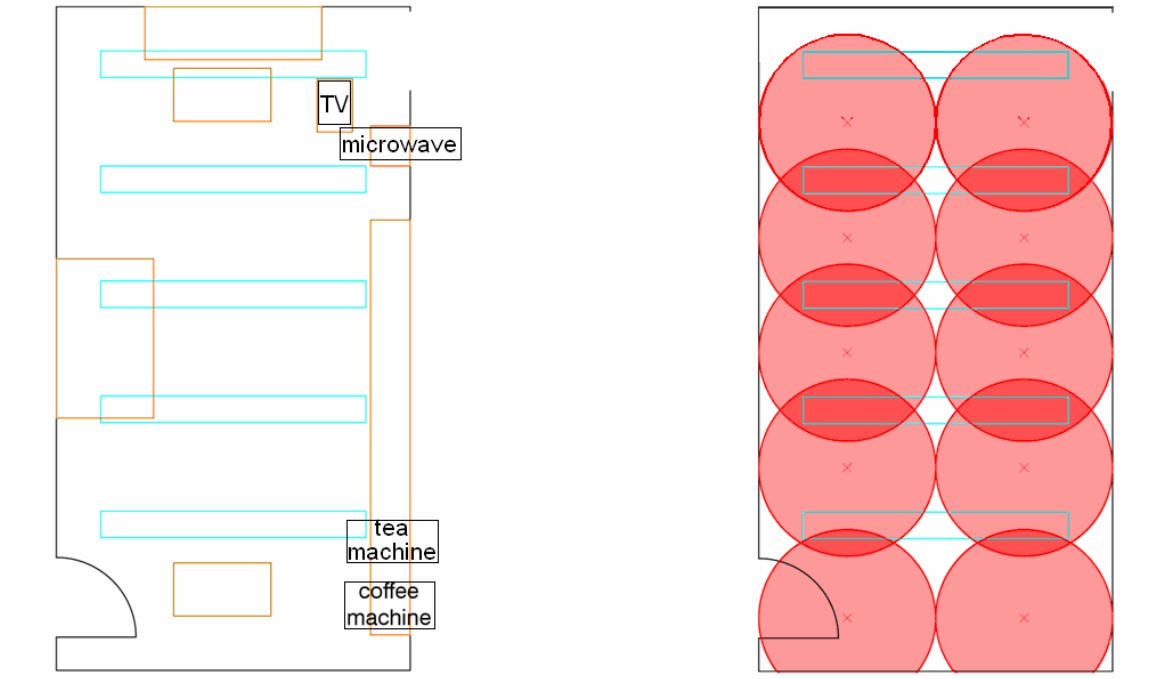


Figure 3.1: Coffee room plans

Figure 3.2: Ceiling of the coffee room with some motion detectors

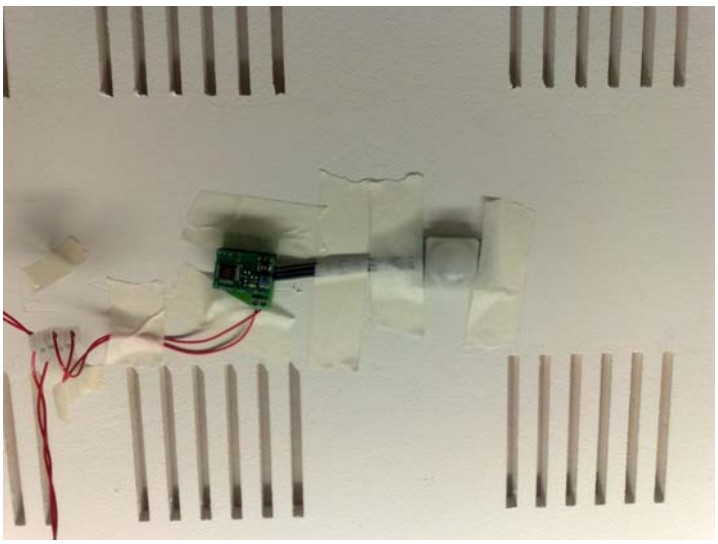


Figure 3.3: Detail of a motion sensor

3.2.1 Detecting peoples positioning

The network should detect all motion in the room so, according to the size of the room and the characteristics of the motion detectors used, it was decided to place the devices as shown in the Figure 3.1. Ten IR motion detectors were placed on the ceiling between the lamps and the ventilation and lamp beams (horizontal rectangles in the left map), providing a circular area of detection with a diameter of 220cm (Right map of the Figure 3.1.

The method of peoples location detection is based on the activation of the specific motion detectors.

3.2.2 Use of electrical devices

A current meter (explained in the section 3.3) was designed and placed between the electrical device and the electrical outlet in the wall. I works also as a cable extension

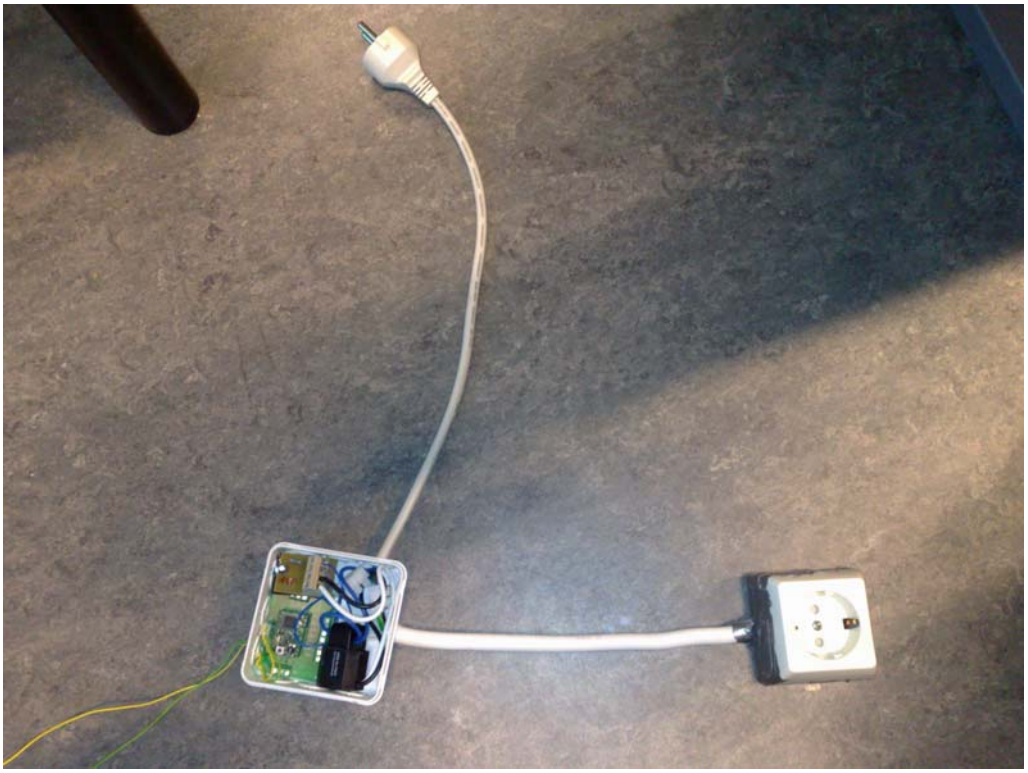


Figure 3.4: Current meter

These current sensors give the system a 12-bit value of the current that the electrical device is consuming.

The devices being monitored are:

- A coffee machine
- A hot water kettle (tea machine)
- A microwave oven
- A television

3.2.3 Counting people

This activity is performed using a set of devices:

- Two photoelectric beam detectors were placed on both sides of the coffee room door, so that the application could recognize when somebody crossed the entrance and the direction of the movement.

Figure 3.5: Detail of the photoelectric beam detectors from above

Figure 3.6: View of the door of the test room with the photoelectric beam detectors and the reflectors

- Monitor the environmental conditions of the room using a thermometer and a humidity sensor.

- The system can be designed to process the information retrieved from the rest of the devices. For example: if the count says that there is nobody in the test room, but the coffee machine has been

turned on, or some motion detectors have been activated, it means that there is at least one person inside the room.

3.2.4 Changes on the environmental conditions into the room

A measuring device with a thermometer and a humidity sensor was installed in the room (Figure 3.7).

Figure 3.7: Air temperature and humidity measuring device

3.3 Design of Linet-compatible current-meters

A current meter was designed to detect the use of various electrical devices. The meter needs to be compatible with Linet nodes and it must be able to measure different levels of electrical consumption.

The idea was to put the meter between the electrical device supply cable and the wall outlet. The Linet compatible current meter works as an extension cable. It does not disturb the normal use of the device being monitored.

The Linet-compatible current meter consists of a 1:3000 Split-Core current transformer and a rectifier. The transformer measures the current of one phase cable or the neutral cable. The voltage output of the current transformer is rectified by a complete-wave rectifier and the resulting voltage is filtered using a RC filter.

Figure 3.8: Detail of the current meter

In order to be able to measure a wide range of device consumption there are two ways to adjust the output voltage. The output voltage has to be less than the maximum Linet-node “analogic input” voltage and high enough to polarize the diodes so as to get a value higher than zero.

1) Changing the RC filter resistor

2) Choosing the number of turns of phase or neutral wire around the Split-Core current transformer.

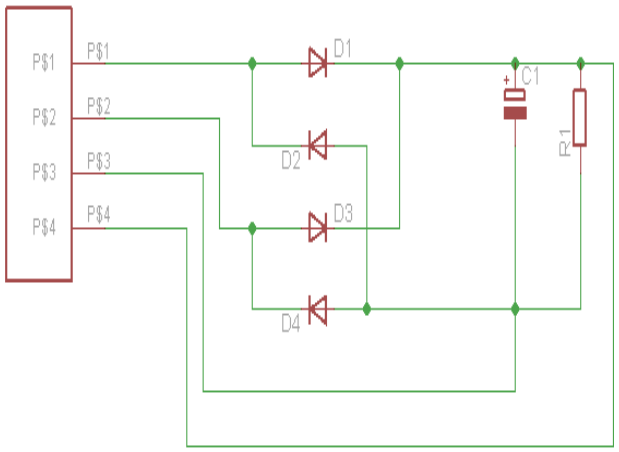


Figure 3.9: Rectifier Schematic

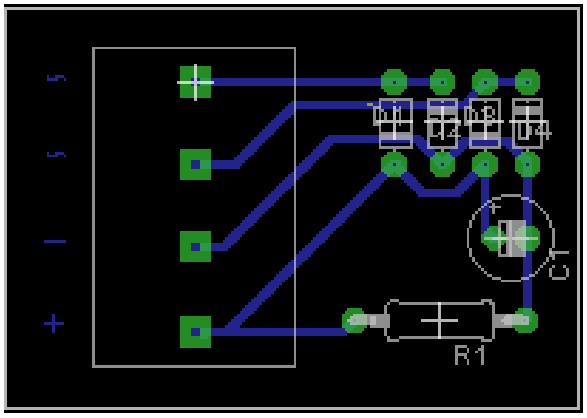


Figure 3.10: Rectifier Board

3.4 Linet network configuration

The network is composed of several basic sensors for monitoring a person's activity. Next table shows a list of sensors that are used in the test system.

Group	Name	Type
1	Bathroom 1 occupation sensor	I/O node
2	Bathroom 1 occupation sensor	I/O node
3	Motion sensor in mailbox	I/O node
4	Temperature sensor in corridor	AD/state
5	Presence sensor in mailbox	I/O node
6	Mailbox lid opening	I/O node
7	Coffee machine working	I/O node
8	Coffee machine brewing	I/O node
9	IR sensor 1	I/O node
10	IR sensor 2	I/O node
11	IR sensor 3	I/O node
12	IR sensor 4	I/O node
13	IR sensor 5	I/O node
14	IR sensor 76	I/O node
15	IR sensor 7	I/O node
16	IR sensor 8	I/O node
17	IR sensor 9	I/O node
18	IR sensor 10	I/O node
19	Photoelectric inside	I/O node
20	Photoelectric outside	I/O node
21	Tea machine current	AD/state
22	Microwave current	AD/state
23	TV current	AD/state
24	Temperature coffee room	AD/state
25	Humidity coffee room	AD/state
26	Coffee machine current	AD/state

Table 3.1: Configuration of the Linet Network groups

There are some sensors in a mailbox located in a corridor and some more sensors in men's bathrooms to monitor the vacancy. The core of the sensor network for activity monitoring test is in the test room.

3.5 Design of a network simulator

With this new application it is possible to design a new network configuration and test it without the real network. The developer will be capable of checking his application by triggering the sensors virtually, using a graphical interface.

The network configuration is read from a XML properties file. The graphical interface shows all nodes. The current version includes the implementation of the "I/O Node" and the "AD State" node configuration. Their components are buttons and sliders respectively.

The simulator connects directly with the oFMS server. By interacting with the buttons and sliders it is possible to change the state of the nodes in the XML Database in the server.

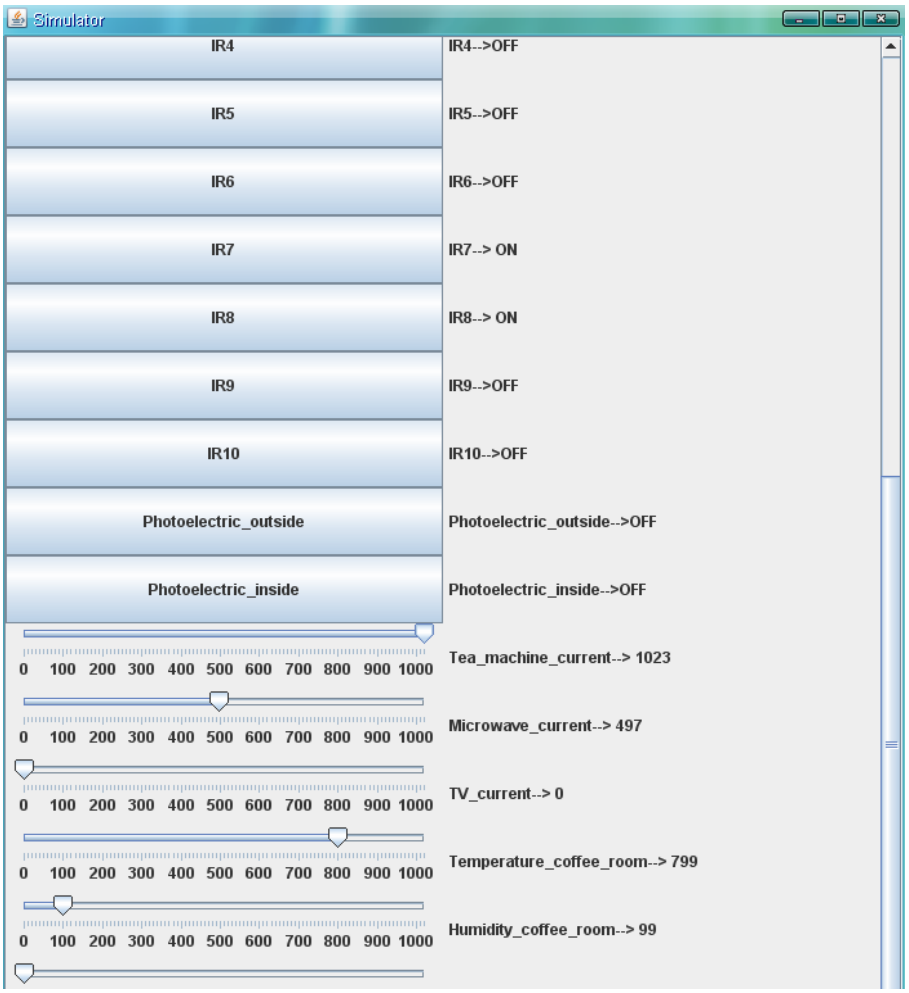


Figure 3.11: Screenshot of the network Simulator

3.6 Improvement of the Home Services application. Activities Recognition Web Service (before known as eCaregiver Web Service)

This application has been developed and explained by Pablo Ramos in his Thesis [Ra07]. Because have been made many changes in several parts, an update of the description is needed.

3.6.1 Architecture

The Activities Recognition WS application is divided to 5 modules that are showed in the figure 3.12:

- Graphical User Interface (GUI)
- Logic module
- Networking module
- Database module
- Activity Recognition module.

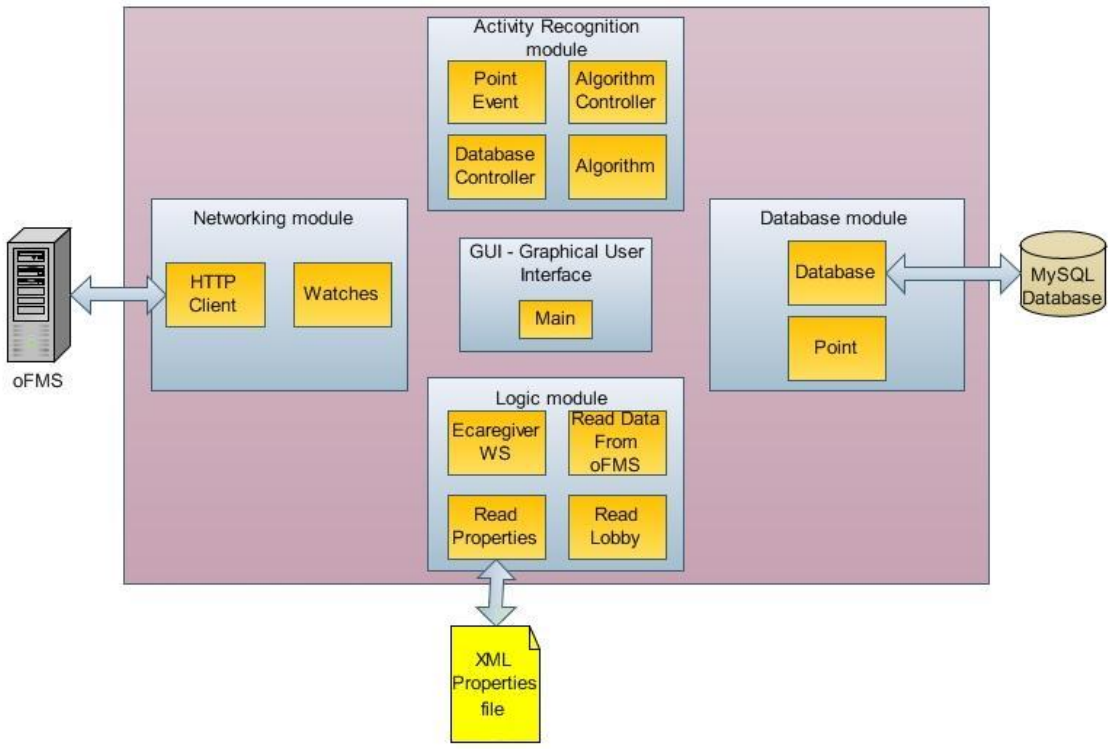


Figure 3.12: "Activities Recognition WS" architecture

3.6.1.1 Graphical User Interface (GUI)

This module has the “main” function that is used to start the Activities Recognition WS application. It was designed for showing graphically all the results of the application, but in this version the graphical interface has been moved to the Web User Interface (WUI). Now it only shows a window asking if it should continue with the same database or on the contrary to reset the database, erase all the tables and entries and start the collection of new data in this point.

3.6.1.2 Networking module

This module was not changed from the last version and this is transcribed mainly from the section 6.2.1 of the Pablo Ramos’ thesis [Ra07]:

<< In addition to a standard way for representing the data, we need means to transfer it through the network. Therefore an abstract request/response model is specified. oBIX specification version 1.0 defines also SOAP and HTTP REST bindings that implement the model and can be used for communication. This gives the manufacturer freedom to use either SOAP or HTTP. “Of course this can also cause situations where a client supports only the one binding model that is not supported by the server” [Jä07].>> [Ra07]

<<Because oBIX is using traditional client-server architecture where clients do not have to implement web servers or expose an IP address, clients always have to poll for changes in the server. “However, polling is not that elegant way to achieve the requirement of the near-real-time information access. Still this is implemented well in oBIX and a model for client polled eventing called Watches is provided. It decreases the amount of the unnecessary network requests and eliminates the resending of the unchanged data” [Jä07]. Watches class implemented this service (see Networking module on Figure. 3.12) and takes care of all the logic behind watches. WatchPoller thread makes possible for clients to implemented changes in the object without having to read the whole object in the oFMS.>> [Ra07]

<<eCaregiver WS (eCaregiver Web Service) module is a HTTP client and the Networking module (referred to as Networking on Figure. 3.12) allows to read, write and invoke to communicate with oFMS. Every method takes an address and a request string as argument and returns the response string.>> [Ra07]

- *<<Read request is used to read any object which has an address (URI). This means an object that has an href.>>* [Ra07]

- *<<Write request is used to change any object that has a href and writable tag that it is true. eCaregiver WS can thus modify its own data on the server to update it according to the current state. eCaregiver WS also uses write request to change the state of the Linet network.>>* [Ra07]

- *<< Invoke request is used to invoke any operation object. eCaregiver WS makes a new watch with an invoke request and adds its own data to it. That is*

how the eCaregiver WS knows if it should change its state. This happens when there are some changes in the Linet network. To see if changes have been occurred, clients poll the watch URI using pollChanges operation. >>[Ra07]

3.6.1.3 Logic module

This module was not changed either from the last version and this is transcribed mainly from the section 6.2.2 of the Pablo Ramos thesis [Ra07]:

<< The Logic module handles the commands coming from the Networking module and Activity Recognition module. ECaregiverWS constructs a session where URI specifies the lobby URL of a obix server (For example lobby typically looks like "http://<hostname>:8080/obix"). ReadProperties sub-module reads general settings of system from an XML file. ReadLobby sub-module reads the Lobby object to find out addresses of the important objects and services. After this module can use the basic functionalities of the oFMS by using these addresses. Finally, the ReadDataFromOfms sub-module recovers all the data from the server, the data is sent from the server to the Activity Recognition module were a database is being created with the new information. >>[Ra07]

<<On the other hand, when starting the program, you have to give the oFMS address as a parameter. Thus the oFMS server does not have to be located at the same computer with the eCaregiver WS. Communication initialization of eCaregiver WS is described in the figure 3.13. >>[Ra07]

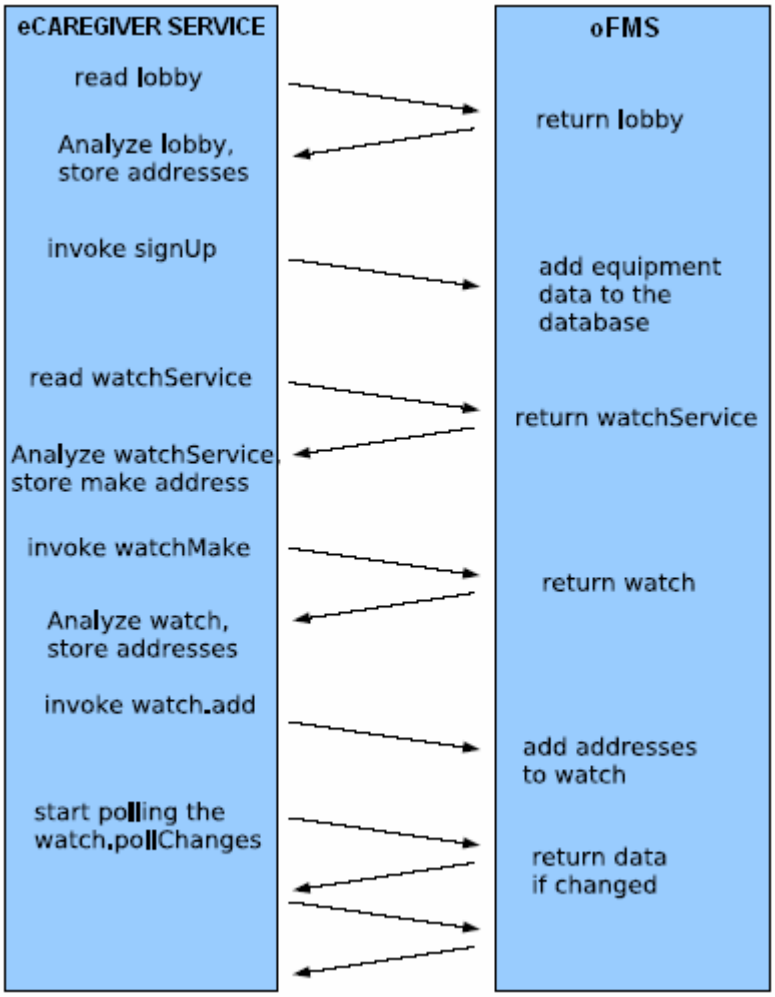


Figure 3.13: eCaregiver Web Service initializing communication with the oFMS [Jä07]

3.6.1.4 Activity Recognition

The Activity Recognition module of figure 3.12 is the brain of the "Activities Recognition WS". This module handles the packets received from the oFMS, takes care of the processing of the packets and of the storing of the results in the database. Activity Recognition module also takes care of the initializing of the database.

The DatabaseController sub-module establishes connection with MySQL server through the Database module and creates the tables in the MySQL database. The tables that need to be initialized are filled with the appropriate initial values. These functions are only called when, in the start of the application, the "Reset database" option is marked.

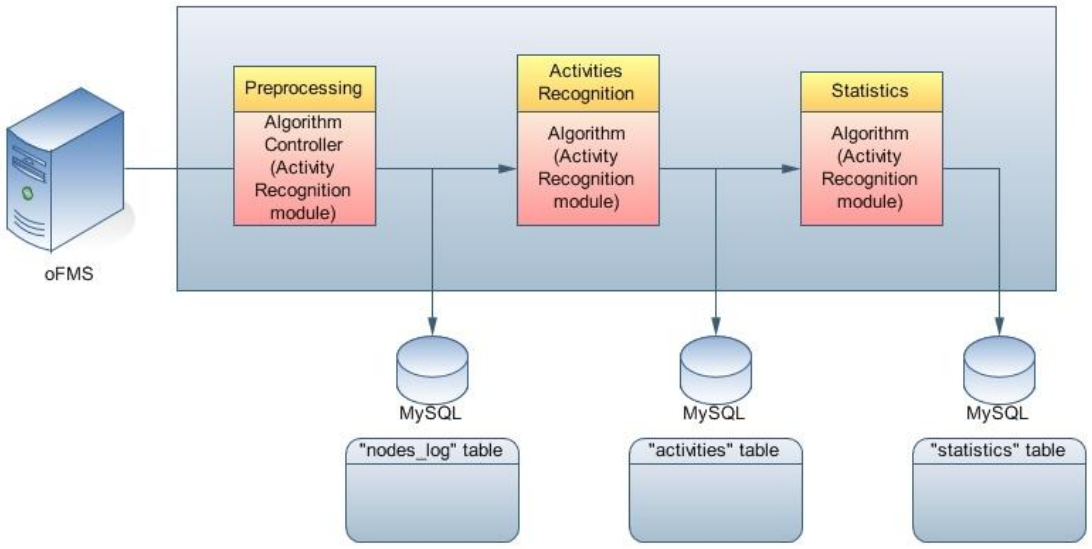


Figure 3.14: Schematic of data analysis

The "AlgorithmController" sub-module takes care of all the logic behind Algorithm module. The data from the oFMS is received here when the “watchPoller” registers change in the Linet network. This data is preprocessed here to get the best values for later analysis. These values are sent to the Algorithm module. Every time the AlgorithmController sends one value to the Algorithm module one row is added to the “nodes_log” table of the MySQL database. The MySQL database design and their tables are explained in the section 3.9.

The preprocessing is necessary because sometimes the real network does not send the correct for the system. For example the “AD State” node gets an analog input and converts it to a 12 bits value, but this kind of nodes does not send any packet when it gets a zero volt input). The preprocessing is also necessary because it is for filtering the noise in the obtained values. For example the “AD State” nodes used for measuring the temperature, the humidity or the electrical current have their state always changing because of their high accuracy and because of the noise. The values obtained by these devices are filtered by calculating the average of last 10 values and by decreasing their accuracy to 10 bits. The filtering is really important because the unfiltered data brings many problems in long term. The “nodes_log” table is got too big and this slows down the system. The efficiency of the system would be reduced considerably, thus powerful computers would be needed that would need much more time and memory for processing.

The "Algorithm" sub-module implements the algorithm for detection of activities and the algorithm for recognition of behavioral patterns, both described in the section 3.7. In this second version of ASED only the activity recognition algorithm has been developed, but the structure is prepared for the behavioral recognition algorithm. Some statistics functions have been developed for testing some long term data. These functions are located in the behavioral recognition block. As is showed in the figure 3.14 when one activity is detected, the Algorithm sub-module adds one row to the “activities” table of

the database and when the system gets the statistics of a defined period, this information will be stored in the “statistics” table. The “Statistics” sub-module is still not developed but the others are working.

3.6.1.5 Database module

The "Database" module on figure 3.12 handles all actions called by the "Activities Recognition WS" with the MySQL database: the queries, the deletions and creations of tables, the deletions, additions and replacements of rows. This module uses the information of the database stored from the "DatabaseController" sub-module of the "Activity Recognition" module.

This is the only class that uses the JDBC driver. This driver provides many functions for using a MySQL database with a Java system (Note: not all the functions are implemented in the current version of this driver. It was detected that the SOURCE function, used for storing the database into an external file, did not work in this ASED system, this function has to be executed from a MySQL console). “On the other hand, Data types of MySQL and Java programming language are not same. It is needed some mechanisms for transferring data between a database using MySQL data types and an application using Java data types. It is needed to provide Java mappings for the common MySQL data types. It is important to have proper type information to store and retrieve parameters and recover results from MySQL statements. The following table represents the default Java mapping for various common MySQL data types:” [Ra07]

MySQL Type	Java Type
CHAR	String
VARCHAR	String
LONGVARCHAR	String
BIT	Boolean
TYNYINT	Byte
SMALLINT	Short
INTEGER	Int
BIGINT	Long
REAL	Float
FLOAT	Double
DOUBLE	Double
BINARY	byte [ ]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp

Table 3.2: Mapping MySQL Data Types in Java [JD08]

The “Database” module has been provided with many specific functions designed for the "activities recognition" algorithm, complex queries that make lighter the work of the "Algorithm" module.

3.6.2 Documentation

The whole application has been documented using Eclipse. The figure 3.15 shows the API of the application.

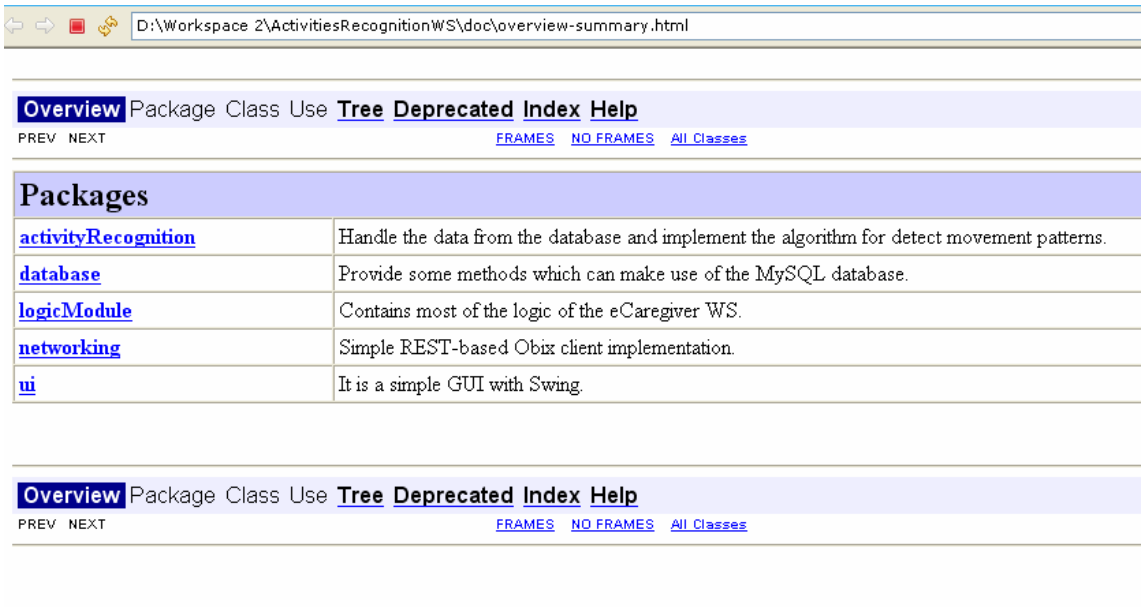


Figure 3.15: "Activities Recognition WS" API

3.7 Design of the algorithms for the activities recognition and for the behavioral pattern recognition

The idea of the ASED system is to do three kinds of analysis with the data collected by the sensor network and preprocessed by the "Algorithm Controller" module. The three analyses are:

- Activities recognition
- Behavioral patterns recognition (not implemented yet)
- Data mining (not implemented yet)

All these analysis should be implemented in the system. The activities recognition analysis was the main point of this thesis, but some steps for the others analyses have been already prepared in the current version.

3.7.1 Activities Recognition

The target of this analysis is to get information about the activities that are happening in real time. First it is necessary to determine the activities to be recognized. Then the corresponding sensor pattern has to be found. Finally the system will try to find these patterns to determine the ongoing activity.

The information necessary for the analysis is provided by the sensors in the format: *sensor ID*, *value* and *timestamp*.

Here are the algorithms that implement the detection of the activities referred in the section 3.1 are implemented in this algorithm module.

3.7.2 Statistics

This analysis is one first approach to the behavioral patterns recognition. Here, some parameters are studied in long term and statistic information about them is obtained and stored in the part of the database designed for it.

- Amount of movement detected each hour and each day.
- Number of coffees during the day.
- Areas of major movement.

3.7.3 Behavioral Patterns Recognition

In this analysis the system should look for long time range patterns. It should monitor variables and their relations between them and to get a trend of these incidences.

Some examples of behavioral patterns that the system could monitor:

- More people come to the coffee room if when the coffee machine is started.
- How many coffees and teas are prepared each day and the hours of most coffee/tea activity.
- The relation between the temperature, the number of persons in the room, the hour of the day, etc.…

3.7.4 Data Mining

Data mining is the process of sorting through large amounts of data and picking out relevant information. It is used to extract information from the enormous data sets generated by modern experimental and observational methods. [DM08]

The using of some data mining techniques could be useful for having efficient methods for getting patterns from the collected data because it is quite hard to do manually for a little network and impossible for a medium or large network. Also, one of the requirements listed in the chapter 1 for having a good solution to elderly and disabled care is that the system should be adaptable and flexible. The data mining is really useful for this purpose because it could automatically get the behavioral patterns of one person and detect its changes.

3.8 Design of an Algorithms Tester application

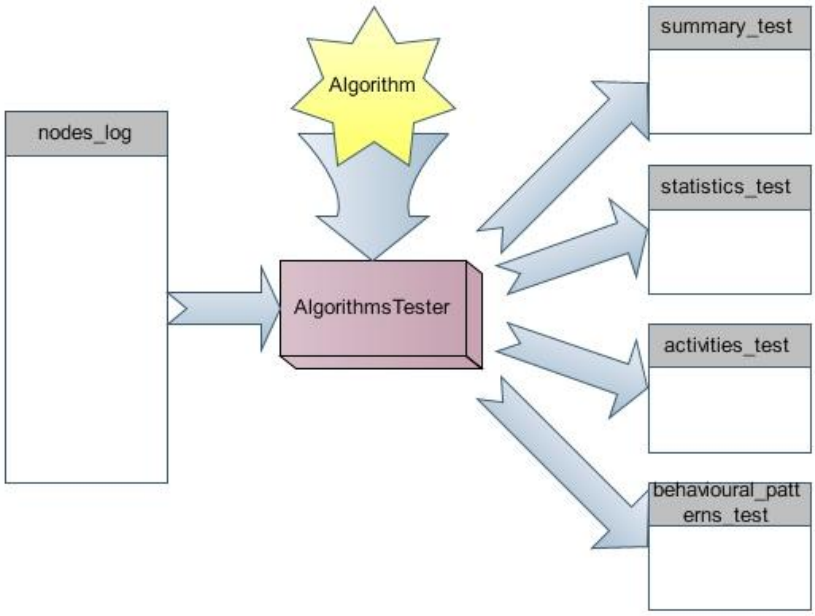


Figure 3.16: Algorithms tester diagram

This application mainly works like the Algorithm module of the Activities Recognition WS (Figures 3.12 and 3.14). The difference is that instead of working in real time and receiving the events from the Algorithm Controller sub-module when they happen the application works offline.

The input of this application are the "nodes_log" entries (“nodes_log” is a table of the database, it is explained in the section 3.9). These rows have been preprocessed in the Algorithm Controller sub-module.

The outputs are the entries of the rest of the tables of the database after processed by the algorithm included in the Algorithms Tester application. These results will indicate if the algorithm tested is suitable for the scenario which is being monitored.

One of the advantages of this tool is that it can compare two different algorithms using the same inputs.

The Algorithms Tester does not disturb the working of the main system because it uses the same table for input, but the output tables can be configured to be different by just changing the name of the table.

3.9 Design of the application Database

The new database has been redesigned in order to better classify the information and to ease the subsequent adding of functionalities and the reconfiguration of the system.

3.9.1 The “Nodes_Log” table

In this table the system stores all the changes of the states of the nodes.

The fields of this table are:

- **Sensor ID**: the identification of the sensor, this number is obtained from the Linet controller and is defined at the configuration of the network.
- **Name**: this name is only stored for the easy management of the database. Each name is stored and linked to its respective sensorID in the XML Properties file.
- **Value**: in this field is stored the status of the sensor at this specific moment.
- **Timestamp**: in this column is written the date and time when the status of this node changed
- **Type**: here is stored the kind of sensor configuration the node is supporting. All the possible configurations are explained in the Appendix B.

3.9.2 The “Corridor_summary” table

The main purpose of this table is to optimize the queries made by the web application in order to show on a web page in near-real time the status of the observed area. Instead of making complex queries to the big log table every time the web browser refreshes the page (2 times per second), this table is updated only when it is necessary and the servlet only has to query simple and directly to this small table.

The fields of this table are:

- **Field**: the name of the variable we are observing.
- **Value**: this field is a string (character array) where we write the status of the variable observed.

The adding of a new functionality is as easy as adding a new row with the name of the target variable

3.9.3 The “Coffeeroom_activities” table

The activities the system has detected are stored in this table.

The fields in this table are:

- **Activity**: the name of the activity detected.

- **Timestamp**: date and time when the activity was detected.

3.10Web User Interface (WUI)

The Web User Interface is a web application that shows graphically information about system and activities through a web browser interface.

The advantages of having a graphical Internet interface are:

- The ability to access information from any computer connected to the Internet.
- The possibility to use the hundred of tools for designing the graphics of web pages, such as Adobe Dreamweaver, Google Charts, etc.
- Only a browser is required.
- Most browsers notify the user if a plugging for the images visualization of the images is needed.

The WUI designed is a web application that consists of some “JSP”s located in an Apache Tomcat server. The web application is written in Java Server Pages, JSP, language because a dynamic update of the data is needed and HTML does not support any service of dynamic changes of the variables of the code. The WUI designed consists of some JSPs located in an Apache Tomcat server. The WUI uses the graphical functionalities of HTML and additional intelligent Java code. It uses the graphical functionalities that HTML gives and it is provided with intelligence with Java code.

JSP was introduced as a way to separate the content from the presentation of the content. “A JSP page is typically an HTML page with special tags for including Java code. The page dynamically compiles into a servlet behind the scenes and executes as such. This makes it possible to write pure HTML (and use HTML tools) without regard to the Java code in the page. There are many ways to further separate content from presentation using both servlets and JSP pages” [Ja08].

The webpage is divided into two pieces:

- Status of the system
- Activities detected

Both pages work with auto-refreshing of two times per second. In the beginning there were some problems with the continuous refreshing. The browser had to load all the pages all the time, including the title and the fixed images. In addition it was impossible to manage the web address bar when the web page was running. This problem was solved using iframes, because it gives the possibility to introduce a HTML document inside an other HTML document. All the contents were divided into two groups: fixed and variable. Each group was put into a different JSP document with the variable part

embedded in the fixed part. Thus the browser only has to load the fixed contents once and the refreshing system is only applied to the variable contents.

3.10.1 Status of the system

Here it is possible to watch the status of certain characteristics of the corridor that can be monitored with the devices of the network designed for this project. These characteristics are:

The temperature in the corridor: we have one thermometer inside the mailbox.

The vacancy status of the toilets: each toilet is equipped with a contact detector to show when the door is locked.

How many letters are in the mailbox, when was the last collection or when was the last collection.

The figures 3.17 and 3.18 are two screenshots of the WUI at two different moments.

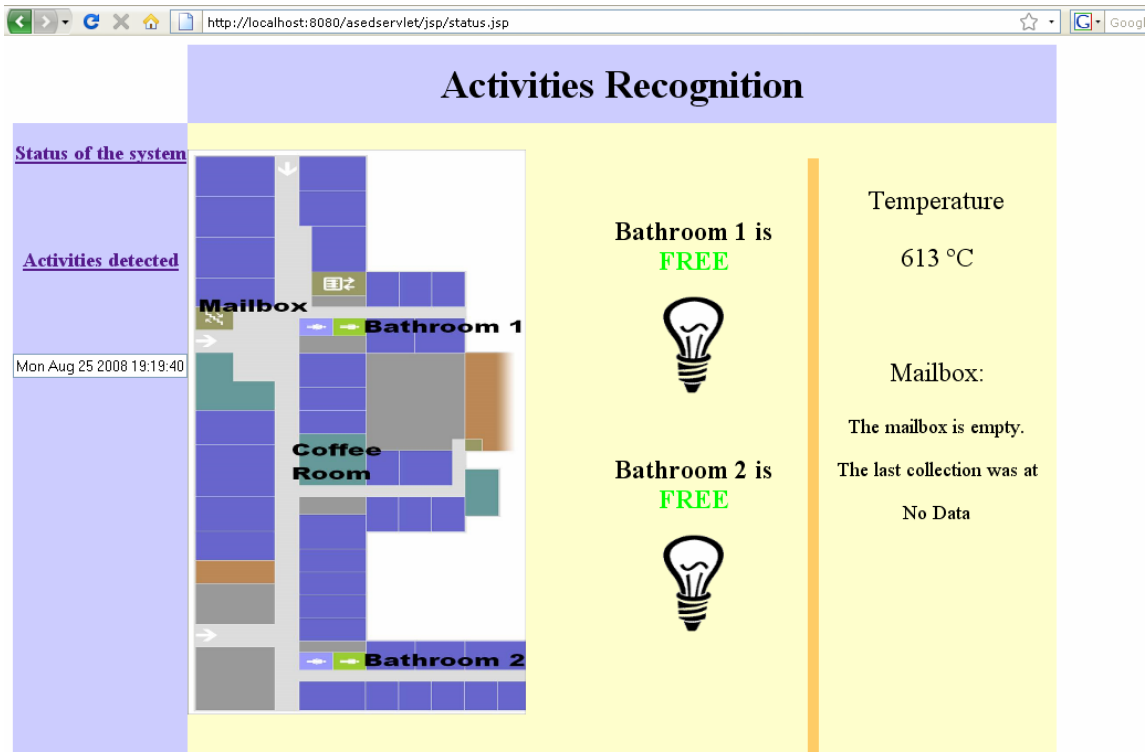


Figure 3.17: Web application screenshot 1. Status of the system

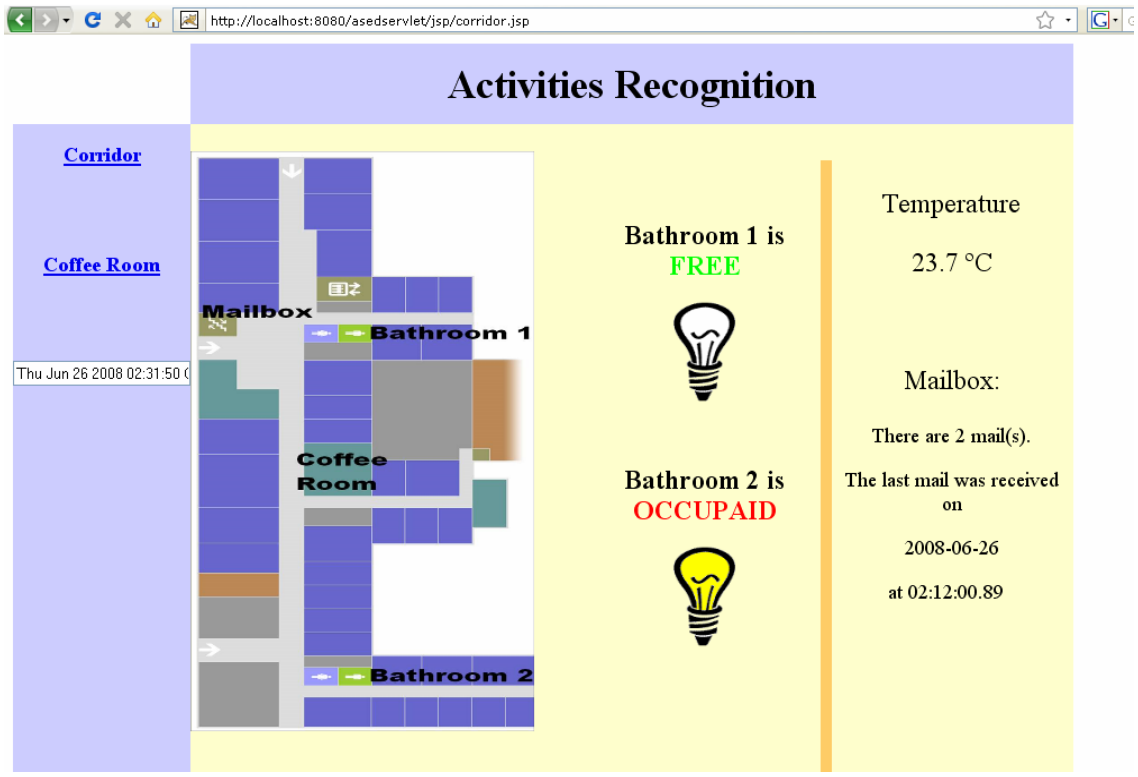


Figure 3.18: Web application screenshot 2. Status of the system

3.10.2 Activities detected

This webpage will work like the corridor_status webpage

This JSP file shows some information of the coffee room: a list of the last activities (previously set) that the system has detected.

Figure 3.19 shows one screenshot of the “status of coffee room” page

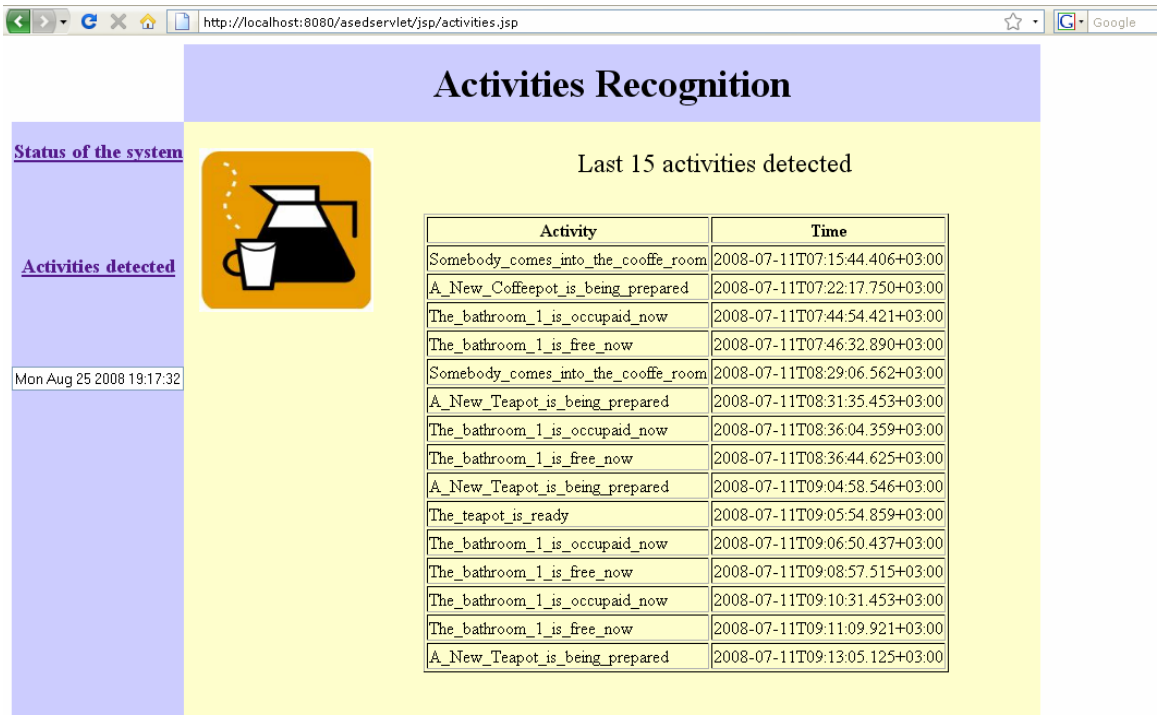


Figure 3.19: Web application screenshot 3. Applications detected

4 Tests and results

4.1 Testing the motion detectors

The system is equipped with motion detectors at model KC7783R. The detectors are tested with different distances (50cm, 1m, 2m, 4m) and with different intensities of motion (standing, walking slowly, walking normally, walking fast, running).

4.1.1 Results

The results in numerical form are shown in the tables 4.1 and 4.2

Total Effectiveness	**82,2%**
Effectiveness when a person is in motion	**94,2%**
Effectiveness when a person is stationary	**26,5%**

Table 4.1: Effectiveness of the motion detectors according to the movement

	In motion	Stationary
Adult (Big)	**98,8%**	**55,0%**
Adult (Normal)	**91,3%**	**30,0%**
Adult (Small)/Child	**92,5%**	**10,0%**

Table 4.2: Effectiveness of the motion detectors according to the size of persons

4.1.2 Conclusion

The results for walking (slow, normal or fast) and running are very similar and the results for all distances are alike. So the system only will differentiate when the object is in motion or stationary.

Moreover the detectors were checked with different size people. It was observed that the size is not relevant when people are in motion. But when people are standing the detector is more sensitive to big person movements than small person movements.

The test shows that this kind of sensor is a good option when we want to detect motion, but just motion, because it is not able to detect effectively the presence.

Because the detector does not discriminate between movements at short distances (around 50 cm) and long distances (4 m), it is not suitable for all kinds of applications. For example if the objetive is to detect when somebody is in the direccion of the detector, but only when he is close to the device, probably these detectors will not work well.

4.2 Counting People Tests

This test was done to check if with just the photoelectric beam detectors at the entrance are enough for detecting correctly people coming in and leaving the test room. For testing this functionality, one person came in and left the test room ten times in a row. This process was done 4 times, with 4 different speeds: running, walking fast, walking with a normal speed and walking slow.

4.2.1 Results of the test

The results are shown in the table 4.3.

	Effectiveness		
Speed	Coming in	Leaving	
Running	0%	50%	25,00%
Walking fast	0%	60%	30,00%
Walking normal	10%	90%	50,00%
Walking slow	60%	100%	80,00%
	17,50%	75,00%	Total

Table 4.3: Effectiveness of the photoelectric detectors in the detection of activities

4.2.2 Conclusions of the test

Although the logic of the algorithm for the counting of people was correct, there is a parameter much more important for this activity: the time between packets with the changes in the network, the refresh time.

Two events happened at different times but into the same refresh time will be collected at the moment. Both events will be received by the algorithm at the same time and according to the code, the element that is analyzed first will have an earlier timestamp than the second. It means that if events happen close enough one to the other, the system will not be capable to recognize the order of the events.

Now, the refreshing time is around 300 milliseconds, which is much longer than the time between activations of the photoelectric beam detectors in normal use. The code is written so that if changes in the detectors arrive in the same package, the first to be analyzed is the one on the inside of the door.

In the results it is showed that if the subject does not walk really slowly the system does not recognize that somebody is coming in the room. Instead the system is much more effective when somebody is leaving, but is this false effectiveness (because of the same reason explained before). This thing also explains why many times in the test when somebody entered the room, the system showed that someone was leaving it (it is not showed in the table).

The refreshing time also affects when one device turns on and turns off fast (or vice versa), in this case the system will not notice any change in the state of this device so the algorithm will not receive any data and the activity will

not be recognized. It happened when the subject was running and walking fast and that is why it did not stored all these leavings.

4.3 Tracing people tests

These tests were made to check how the IR motion sensors installed on the ceiling work. We are going to do some tests for answering some questions, which are important for future applications. The following questions need to be answered:

- When there is only one person in the room, is there always at least one sensor activated?
- Is the system capable of tracing one person when he is alone in the room?
- Is the system capable of tracing two persons in a room?
- Is the system capable in tracing more than 2 persons?
- What is the maximum number of motion sensors that are activated simultaneously with one person and with this configuration?
- Is the system capable of working well when the people in the room are at rest? (Seated or standing).

4.3.1 Results of the test

- When there is only one person in the test room, is there always at least one sensor activated?

  No. Many times when there is motion detected by a motion detector the sensors signal is not constant. Their boolean signal changes it state even when there is a continuous movement. Henceforth this problem will be called blinking of the signal. Besides the blinking when the person is not in motion the sensors do not send any signal. These two problems explain why sometimes there is not any detector activated when somebody is in the test room

- Is the system capable to trace 1 person when it is alone in the room?

  No. The current algorithm is not capable of tracing one person because the sensor signal blinks. It could be possible to detect areas of movement by improving the algorithms and by better preprocessing of the data from the motion sensors.

- Is the system capable to trace 2 persons?

  No. There were many problems when trying to trace one person so it is more difficult to trace two persons.

- Is the system capable to trace more than 2 persons?

  No. There were many problems when trying to trace one person so it is more difficult to trace more than two persons.

- What is the maximum number of motion sensors that can be activated at once with one person and with this configuration?

  The maximum number of motion detectors turned on at the same time because of a person standing was six. This test was not performed with persons moving around in the room.

- Is the system capable of working well when the people in the room are at rest? (Seated or standing).

  No. As seen in the test 1 in section 4.1 these detectors do not work well when the people are not moving.

4.3.2 Conclusions of the test

The motion sensors are quite difficult to handle, they need a preprocessing for erasing the blinking and there is also the problem of only work with motion.

Probably the motion detectors are not the best option for tracing and for positioning of people. In the future the system could use the information of other devices for being successful these activities.

But there is other parameter the system could use easily for getting a better understanding of the behavior of the persons in the house: the amount of movement. The amount of movement can be used in a statistic way for the following of the level of activity during the day and for detecting deviations that could show a problem of the monitored person.

4.4 Using electrical devices tests

This test measures the effectiveness of the current meters. In order to test them it was done a long term test and a test where the devices are switched on and switched off a few times.

In the first test the devices are turned on during 5 minutes (the tea machine was less time as a precaution)

In the second test the devices are switched on and switched off 10 times, with periods of 30 seconds between each switching.

4.4.1 Results of the tests

4.4.1.1 Long term test

- Coffee Machine: no problem.

- Tea Machine: no problem.
- Tea Machine: no problem.
- TV: The signal blinks.

4.4.1.2 Switching on/off

The results are shown in the Table 4.4.

	Effectiveness	
	Switchings on	Switchings off
Coffee Machine	100%	100%
Tea Machine	100%	100%
Microwave	100%	100%
TV	100%	100%

Table 4.4: Switching on/off test results

4.4.2 Conclusions of the tests

Here the blinking of the signal of the TV current is produced by the preprocessing. The AD State nodes that are connected to the Linet-compatible current meters have one problem: when they are measuring values greater than 0 volts and receive a 0 volts value does not send any packet with its change of state. This problem was solved in the preprocessing with a timer that sends a 0 state value when the preprocessing module does not receive any value greater than 0 during this period, but this solution is colliding with the filter (also in the preprocessing) of all the AD State nodes (the filter does not send any packet to the analysis recognition module because the current consumed by the TV is very constant). It will be necessary to adjust the filter of this node.

The tests were very successful recognizing the switchings and measuring the high state value so in the future this information can be used in more complex algorithms.

4.5 Test of long term running system

For this test the system has been working continuously for two weeks.

4.5.1 Conclusions of the test

The system has been working correctly during these days, there was not any error nor any exception but as time went on the computer was slowing. It could be because everyday more entries were added to the tables and many of the queries the system does every time it receives a packet from the oFMS are to the biggest table. After 2 weeks the “nodes_log” table had more than 180000

rows and probably the time spent with the queries of a table rises according the size of the table.

It could be a good idea to limit the number of rows the algorithm uses in its queries.

5 Conclusions

This new version of ASED is a basis for the development of more complex algorithms for the activities recognition and for detection of behavioral patterns.

There is still a lot of work to do for getting of a smart monitoring system but the layouts for the developing of it are already designed.

5.1 Problems of the current version

- Refreshing time (time between the receptions of the packets with the changes in the state of the nodes of the network) is too long for the detection of some activities. It will be necessary to change the method for detecting these activities or to reduce the refreshing time.

- It is difficult to get reliable data from the motion detectors. The signals blinking need a better preprocessing. It could also be interesting to test other sensor for detecting presence.

- There are some unsolved problems with the preprocessing of the data from the current meter which measures the current consumed by the TV.

5.2 Future work

After fixing the above mentioned problems the next suggested steps are:

5.2.1 Developing of a Behavioral Patterns Recognition algorithm

The algorithm for detecting behavioral patterns should be capable of detecting long term patterns. These patterns have to be found from real data and transcribed into the algorithm.

5.2.2 Data mining analysis

The system collects much data. This data contain much useful information about the way of life of the people and about their behavioral patterns. The problem is that these characteristics many times are difficult to organized and it is difficult find them. The data mining is the concept of using some special algorithms, mathematical, logical and statistical algorithms in order to find these relationships between all the parameters of the system.

The results of the data mining analysis are a group of objects which are interesting for monitoring. These techniques could give some important relations (not always obvious) between sensor data.

Appendix A: Manual

A.1 Installing ASED on Windows

A native Windows distribution of ASED has been available from this CD. This section describes the process for installing ASED on Windows.

To run ASED on Windows, you need the following:

- A 32-bit Windows operating system such as 9x, Me, NT, 2000, XP or Windows Vista.
- Generally, you should install ASED on Windows using an account that has administrator rights. Otherwise, you may encounter problems with certain operations such as editing the PATH environment variable or accessing the Service Control Manager in MySQL.
- TCP/IP protocol support.
- Enough space on the hard drive to unpack, install, and create the databases in accordance with your requirements (generally a minimum of 200 megabytes is recommended.)
- Your system must be Java SE 6 enabled.
- MySQL must be installed and running (see *MySQL section*)
- Apache Tomcat must be installed and running (see *Apache Tomcat 6.0 section*)
- Linet network must be running. The controller has an Ethernet interface, the IP address and UDP port of the controller can be set with a controller interface. For details, see *Configuration Manual* in `<CD-ROM path>\Linet\Data sheets.`

To install ASED v2.0 manually, follow these steps

1. ASED distribution for Windows can be located in CD. You have to install ASED from a Zip archive. You must install it into the `C:\ASED` directory.
2. Extract the install archive (`<CD-ROM path>\ased.zip`) to the chosen installation location using your preferred Zip archive tool. Some tools may extract the archive to a folder within your chosen installation location. If this occurs, you can move the contents of the subfolder into the chosen installation location.
3. Set the java path in Windows. To set the PATH permanently, add the full path of the `jre1.6.0_07\bin` directory to the PATH variable.

Typically this full path looks something like `C:\Program Files\Java\jre1.6.0_07\bin`. Set the PATH as follows:

Choose Start, Settings, Control Panel, and double-click System. Select the Advanced tab and then Environment Variables. Look for "Path" in the User Variables and System Variables. If you're not sure where to add the path, add it to the right end of the "Path" in the User Variables. A typical value for PATH is: `C:\Program Files\Java\jre1.6.0_02\bin` (Capitalization doesn't matter). Click "Set", "OK" or "Apply".

The new path takes effect in each new Command Prompt window you open after setting the PATH variable.

4. Creating an eCaregiver Database. Open *MySQL Command Line Client* (Start> All Programs> MySQL> MySQL Server 5.0) and click enter when asked for password. Then, you will see a screen as the Figure A.1.

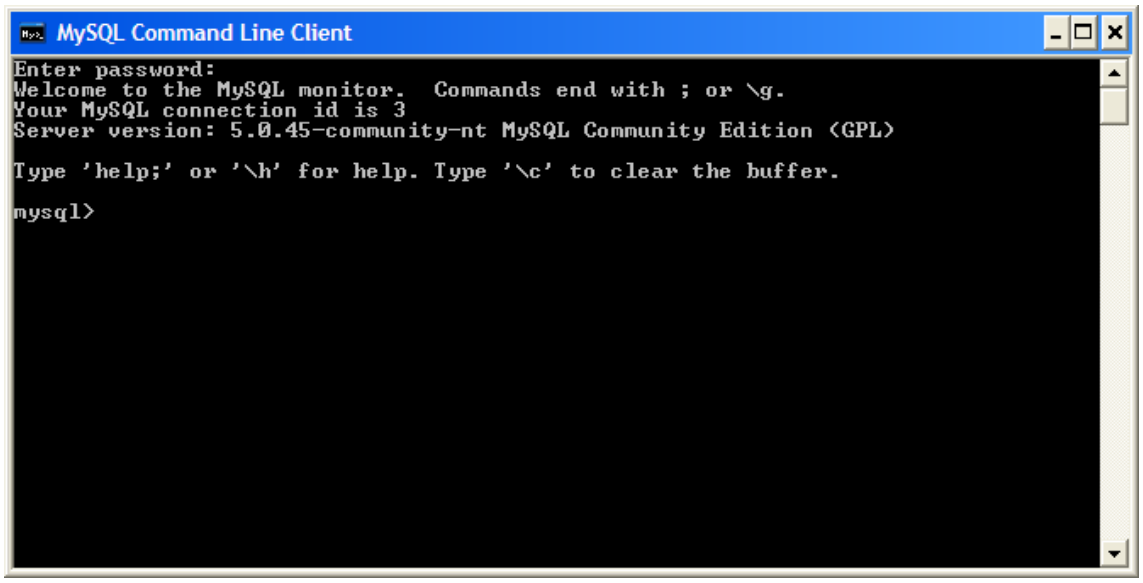


Figure A.1: MySQL Command Line Client

```
mysql> CREATE DATABASE ecaregiverdb;
Query OK, 1 row affected (x sec)
mysql> USE ecaregiverdb;
Database changed
mysql> GRANT ALL ON ecaregiverdb.* TO ecaregiver@localhost
IDENTIFIED by 'boikot81';
Query OK, 0 rows affected (x sec)
mysql> EXIT;
```

Click Start, Run, and enter `<CD-ROM path>\MySQL\Test.bat`. If everything is ok you should see a window similar to the one below:

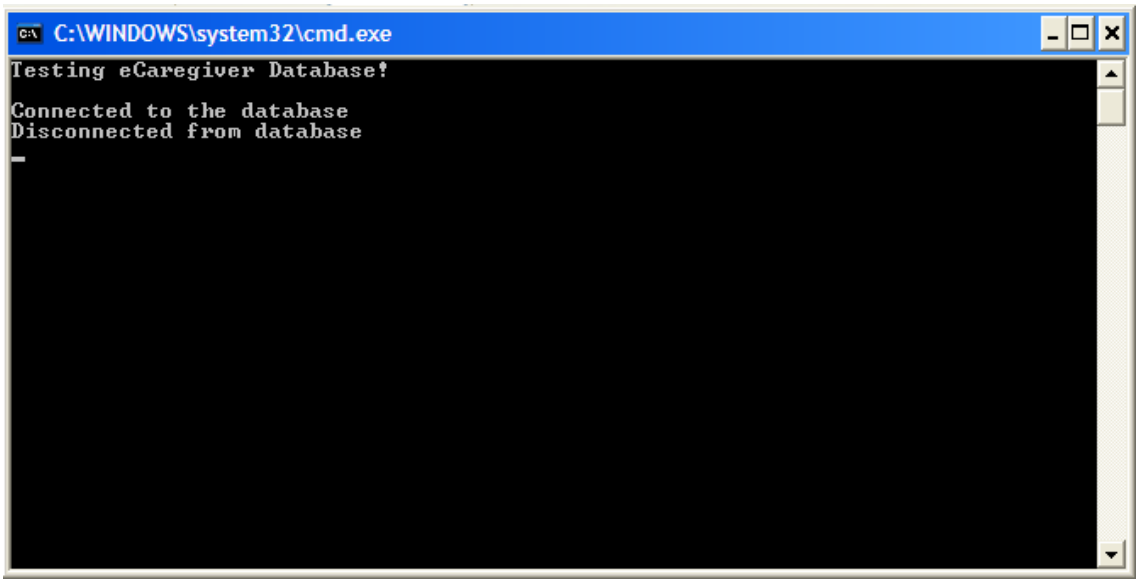


Figure A.2: Database test

In other case, repeat the previous steps and check the files are all there.

5. To start Apache Tomcat 6.0. If you have successfully installed Tomcat, you simply select the *Monitor Tomcat* from Apache Tomcat 6.0 menu from your Windows Start menu.

6. Now that you have installed Tomcat, you have to install the eCaregiverWS web application on Tomcat; so copy folder `asedServlet` from `<CD-ROM path>\Apache Tomcat 6.0\webapps\` to `C:\Program Files\Apache Software Foundation\Tomcat 6.0\webapps`.

7. To check if Linet network is running in port 130.233.120.94. Click Start, Run, and enter `<CD-ROM path>\Linet\Linet Client\Linet.bat`. You should see a screen like the one shown in Figure A.3.[1]

[1] Firewall could cause some problems.

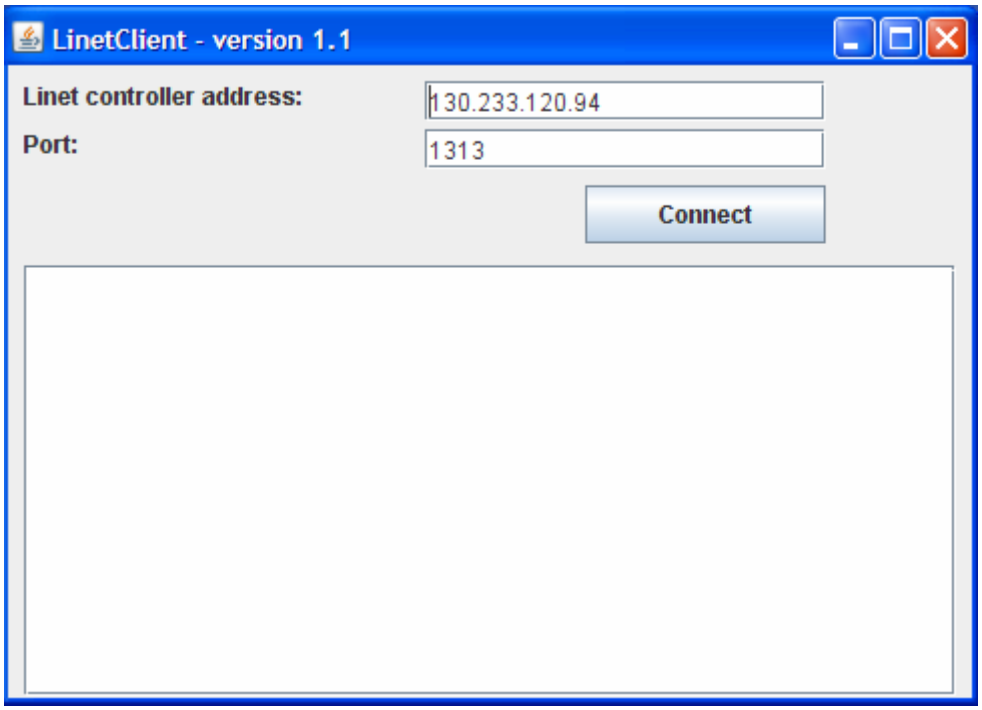


Figure A.3: Linet Client

Click “Connect” and if everything is ok you should see the nodes installed in the network.

8. Testing the oFMS. To start oFMS, simply open the file `oFMS.bat` from the directory `C:\ASED\startUp\` and you should see a screen like the one shown in Figure A.4.

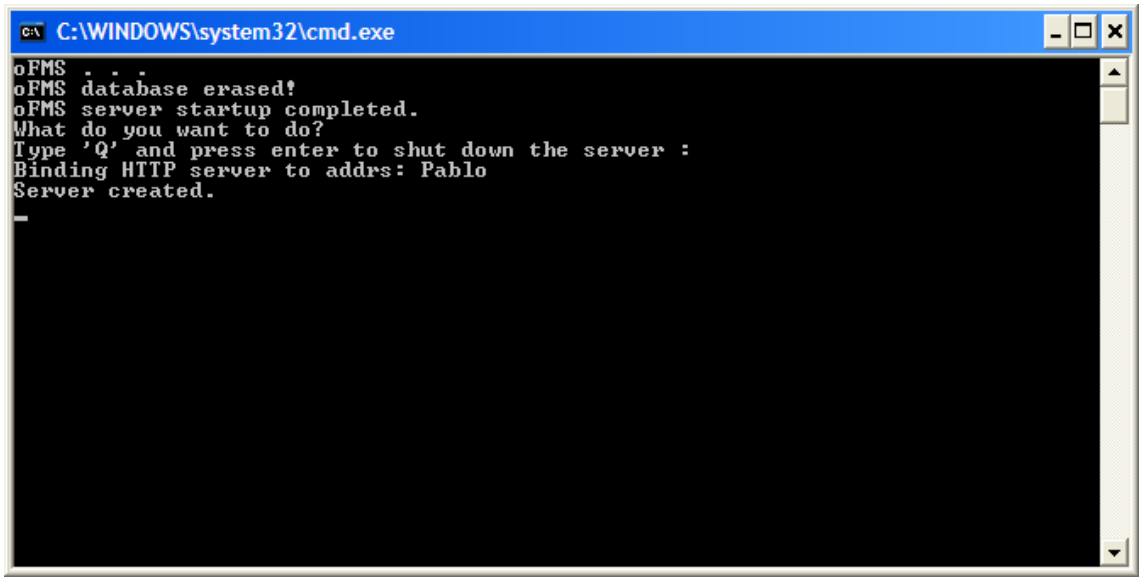


Figure A.4: oFMS screen

By default, oFMS starts on port 8081. Point your browser to http://<hostname or IP address>:8081/ and you should see a screen like the one shown in Figure A.5.

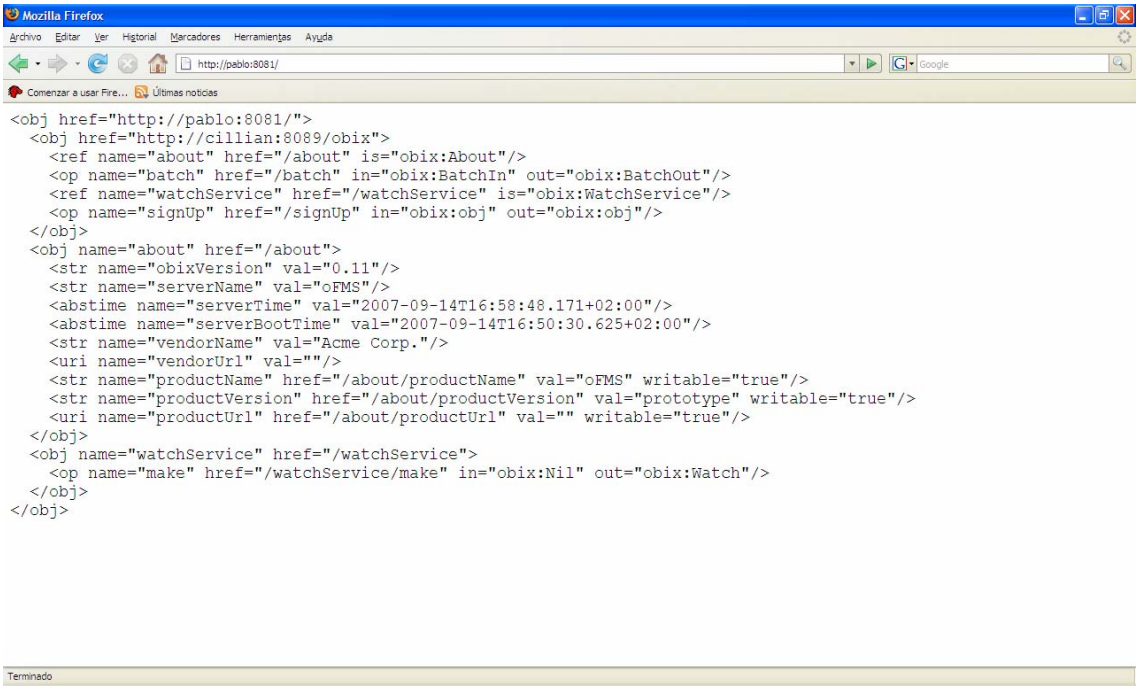


Figure A.57: oFMS server

9. Testing the Linet2oBIX Adapter. To start Linet2oBIX Adapter, simply open the file `Linet2oBIX Adapter.bat` from the directory `C:\ASED\startUp\` (Previously, you have to run oFMS). Point your browser to http://<hostname or IP address>:8081/ and you should see a screen like the one showns in Figure A.6 with information related to all the nodes connected to the Linet network.

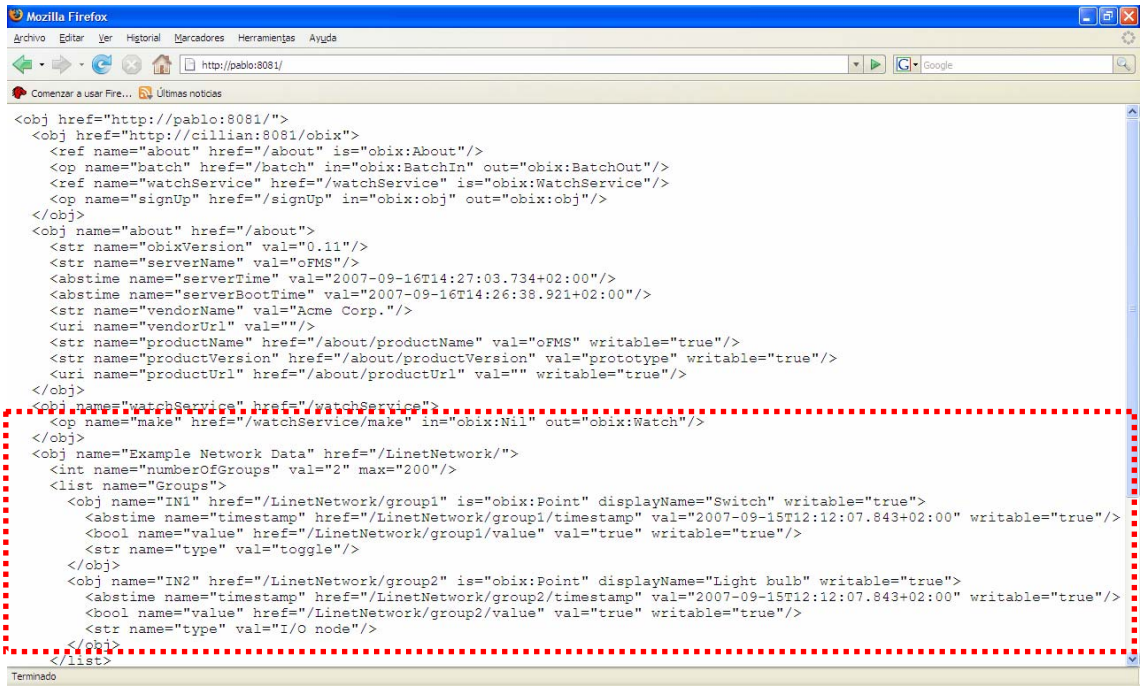


Figure A.6: Linet2oBIX Adapter screen

10. Testing the Activities Recognition WS and the Web User Interface (ASEDservlet). First it is necessary to put the server IP address (IP address of the computer where the web application and the MySQL database are located) in status.jsp and in activities.jsp files. The lines that need to be completed are:

- (status.jsp; Line 33):
  <iframe src=http://IPADDRESSREQUIRED:8080/asedservlet/jsp/status_var.jsp
- (activities.jsp; Line 36):
  <iframe src="http://IPADDRESSREQUIRED:8080/asedservlet/jsp/activities_var.jsp"

In both cases IPADDRESSREQUIRED has to be changed for the real IP of the Tomcat server.

To start Activities Recognition WS, open the file `ActivitiesRecognitionWS.bat` from the directory `C:\ASED\startUp\` (Previously, you have to run oFMS, Linet2oBIX Adapter and Apache Tomcat 6.0)

To start Web Server User Interface, point your browser to http://<hostname or IP address>:8080/asedservlet/jsp/status.jsp and you should see a screen like the one shown in Figure A.7.

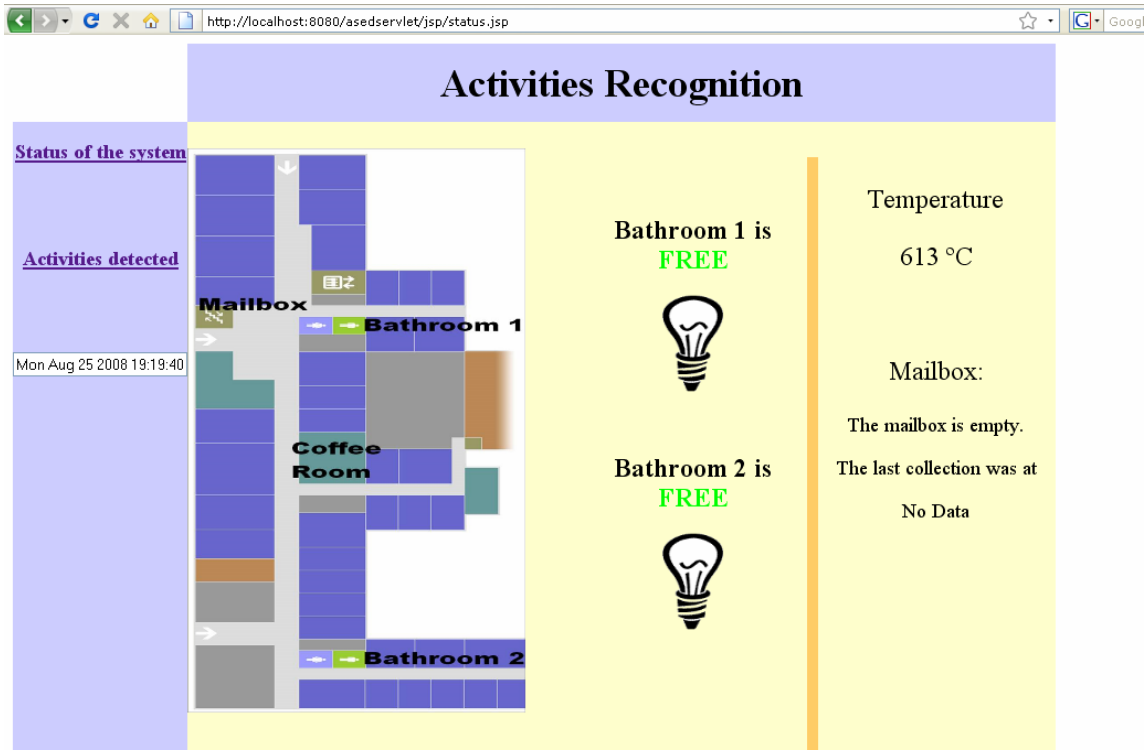


Figure A.7: Web User Interface

If everything is ok you should see the effects in the web page when some nodes change their state (For example the bulb of the Bathroom 1 has to change when the Bathroom_1_occupation_sensor node is activated or deactivated).

Note: if the system does not recognize the jdbc driver (mysql-connector-java-5.0.6-bin.jar) wich is located in the directory:

<*\Apache Software Foundation\Tomcat 6.0\webapps\asedservlet\WEB-INF\lib>

copy this file to the Apache Tomcat lib directory:

<*\Apache Software Foundation\Tomcat 6.0\lib>

A.2 MySQL

A.2.1 Installing MySQL on Windows

Installing MySQL is a uniform process across all versions of Windows. The only difference is that, under later releases (Windows NT, Windows 2000, and Windows XP), it can be installed as a service so that it's always available.

A.2.1.1 Getting the Binary Distribution

You'll find a free download of MySQL Community Server software in the official page: http://dev.mysql.com/downloads/mysql/5.0.html#downloads

A.2.1.2 Running the MySQL Installer

Run it. Accept the default options for all the installation questions, including installing it into C:\Program Files\MYSQL, and let it install.

A.2.1.3 Configure the MySQL Server

First choose *Standard Configuration*. In the next window, choose *Install as Windows Service* and accept the default options. Afterwards, *modify security settings* check box must be disabled and click "OK". Finally, click “Execute”.

A.2.1.4 Differences between MySQL under Windows 95 and 98 and Windows 2000, XP, and NT

Under the Windows 2000, XP, and NT operating systems, MySQL is installed as a service, which means that it is not needed an explicit start after booting. Under Windows 95 and 98, it is needed to run *winmysqladmin* to start MySQL after each reboot.

A.2.2 Tutorial

There are many manuals and tutorials about MySQL. One good online tutorial is available from http://dev.mysql.com/doc/refman/5.0/en/index.html. See Section 3, “Tutorial”.

A.3 Apache Tomcat 6.0

A.3.1 Installing Tomcat under Windows

A.3.1.1 Downloading the Distribution

Point your browser to http://apache.rediris.es/tomcat/tomcat-6/v6.0.16/bin/apache-tomcat-6.0.16.exe. Save the file to a known location.

A.3.1.2 Setup

Installing Tomcat on Windows can be done easily using the Windows installer. Its interface and functionality is similar to other wizard based installers.

When you run the installation program, it first searches for an installed Java Development Kit. If you don't already have a JDK installed, download one from http://java.sun.com and install it before installing Tomcat. After the Tomcat installer finds a JDK, it presents a license agreement.

After agreeing to the license, you can choose what parts of Tomcat you want to install. If you are running Windows NT, 2000, or XP, you may install Tomcat as an NT service. Figure A.8 shows the component selection screen.

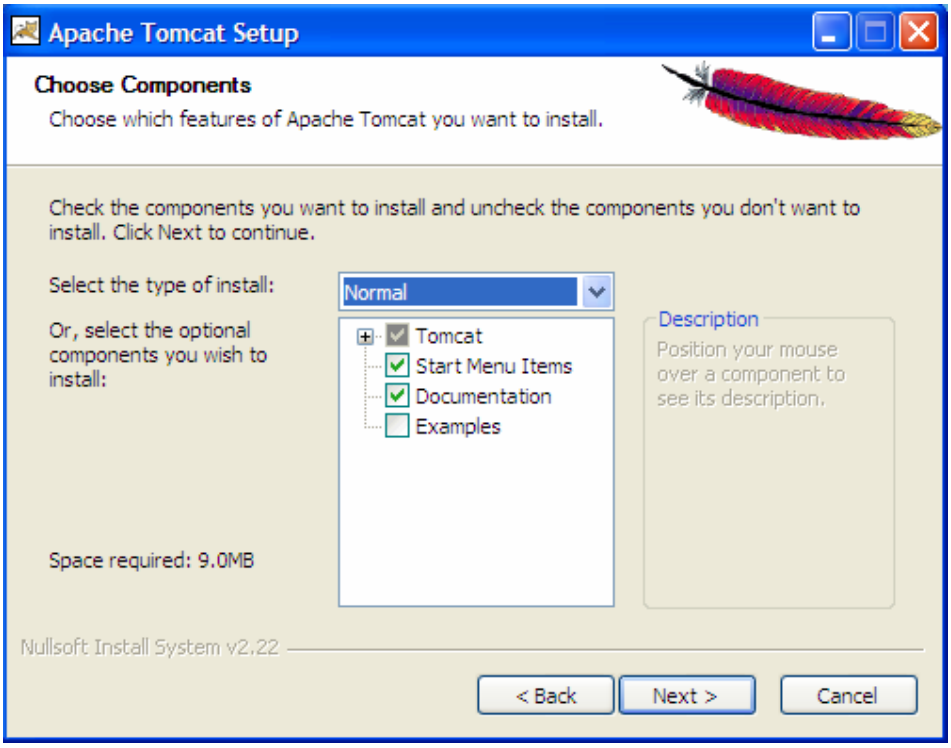


Figure A.8: Selection of components you want to install

After the installation program completes, Tomcat is installed on your system.

A.3.1.3 Testing the Installation

To start Tomcat, simply select the Monitor Tomcat from Apache Tomcat 6.0 menu from your Windows Start menu.

By default, Tomcat starts on port 8080. Open the URL http://localhost:8080 with your favourite web browser. You should see the default Tomcat welcome web page (Figure A.9).

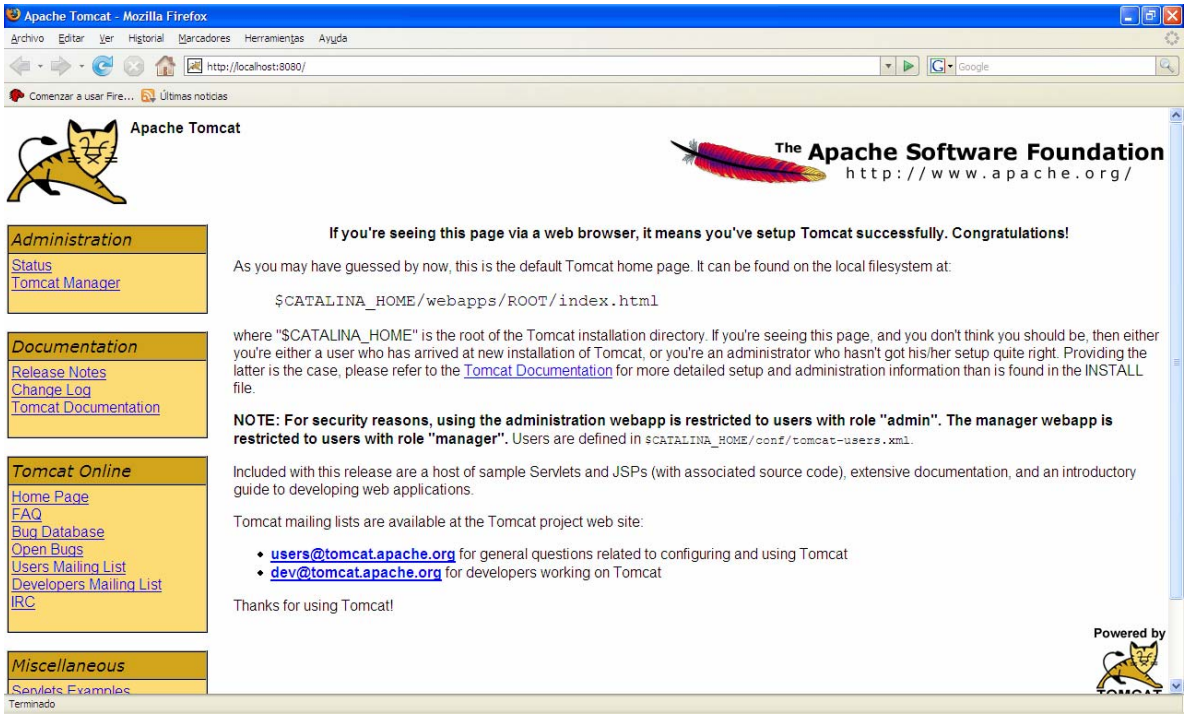


Figure A.9: Apache Tomcat welcome page

A.3.2 Servlet Container Basics

Now that you have installed Tomcat, you have to know how to install your web applications on it. The information in this section applies to any conformant servlet container, not just Tomcat.

Apart from the `bin` directory in which `startup.bat` and `shutdown.bat` live, the most other important directory is the `webapps` directory, which will contain your web applications.

A web application is bundled as a WAR file (for Web Application aRchive), which is just an ordinary JAR file with the .war extension. To deploy a web application, you simply drop its WAR file in the `webapps` directory and presto! Tomcat automatically installs it for you. You know this because Tomcat will create a *subdirectory* of `webapps` with the same name as your web application.

The WAR archive file must contain the following:

- A `WEB-INF` directory, into which you put a file called `web.xml`, used to configure your webapp.

- A `WEB-INF\lib` directory, into which you put all Struts JAR files, including any thirdparty JAR files your application uses.

- A `WEB-INF\classes` directory, into which go the .class files for your webapp.

- Your JSP, HTML, image, CSS, and other files. JSP and HTML files are usually on the "root" of the WAR archive, CSS usually in a styles subdirectory, and images in an images subdirectory.

Figure A.10 illustrates this information.

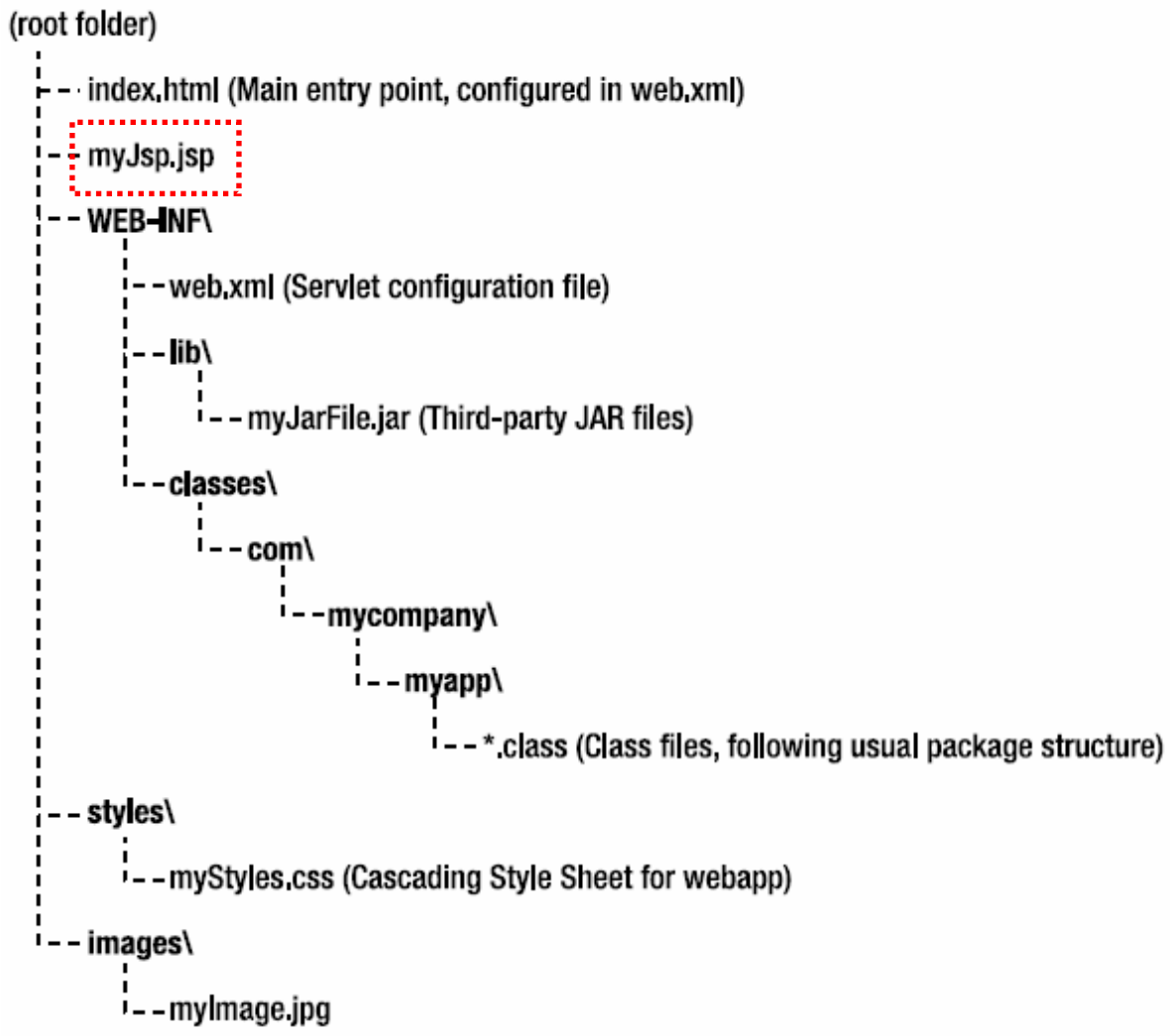


Figure A.10: Structure of a simple WAR file

A.4 Configuring & Using ASED 2.0

A.4.1 Linet Network Controller

In Linet system, the network controller is used as the tool to set up the network. The controller supports two alternative interfaces for this, which are the fixed user interface and the terminal interface. An interface is required during set-up only, so the user may disconnect it when the configuration has been done and the network is running.

These interfaces provide similar procedure to set-up the network but we will choose the following.

A.4.1.1 Configuring. The terminal

The controller has an Ethernet interface, which may be connected to a PC via Ethernet-port. The configuration may then be carried through using any terminal application (as HyperTerminal in Windows), so any Linet-specific software is not required on the PC.

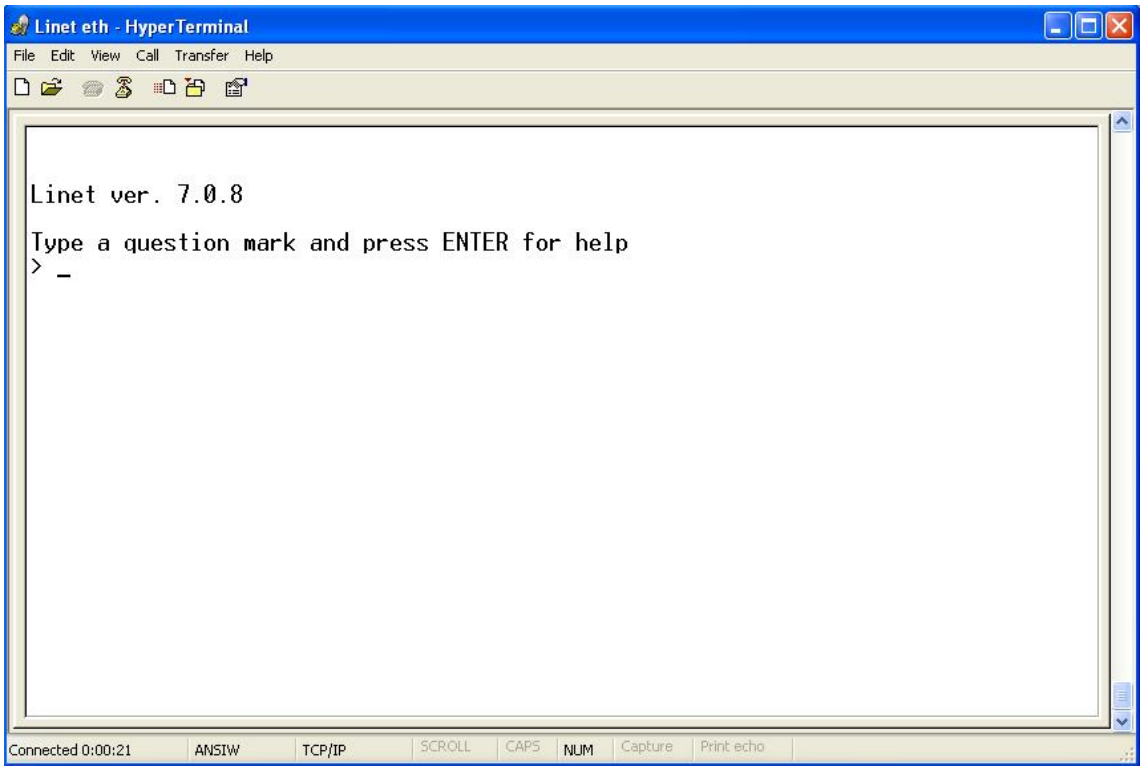


Figure A.11 Start-up screen on the terminal

A.4.2.2 Installing nodes

As example, here it shows how to add two new nodes: one that has a boolean state (I/O Node) and other with a 12 bits state (AD State). These configurations are the configurations used in the network described in this thesis.

When the terminal is used, the function is selected by pressing a number key according to the displayed selection. 'Return' represents 'Ok'-key on the fixed interface, 'Esc' represents 'Cancel'.

By default, the terminal is at command-mode. This is the mode in which the controller accepts run-time control instructions. To enter the configuration mode, the user should type 'config' and press enter. Then the configuration menu appears.

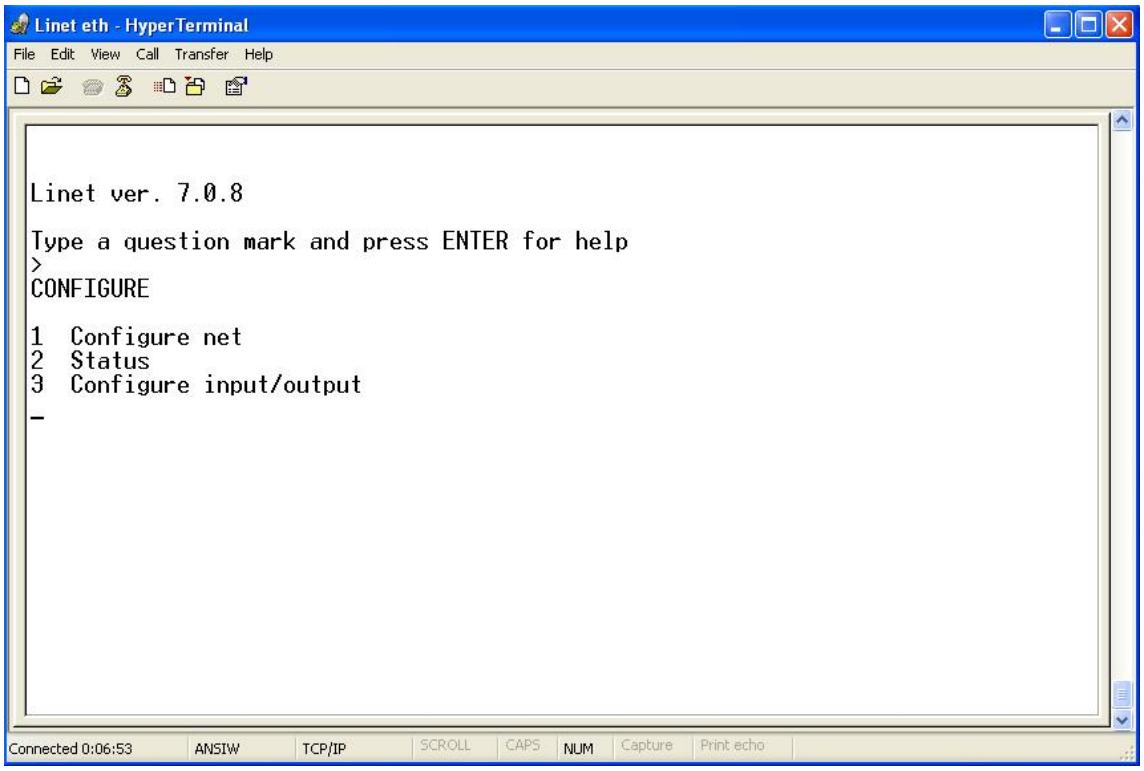


Figure A.12 Configuration menu

To create a new I/O Node group the user selects '1' (configure net)...

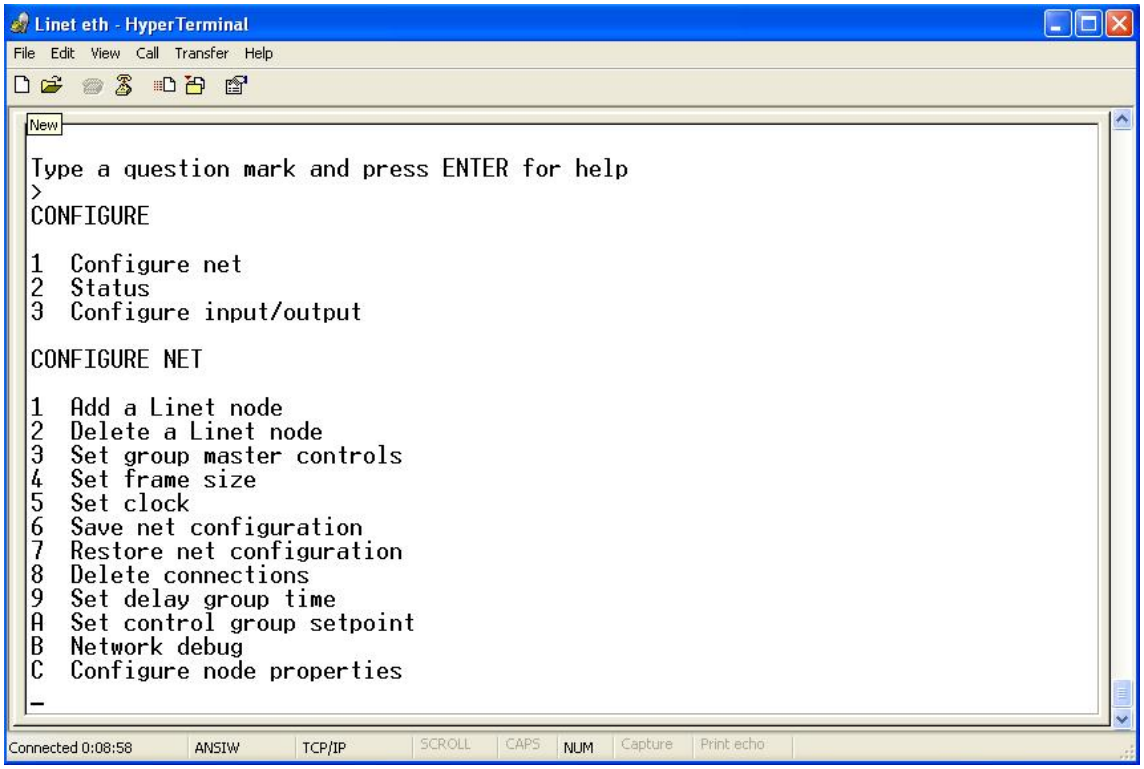


Figure A.13: Configuration net

...then '1' (Add a Linet node)...

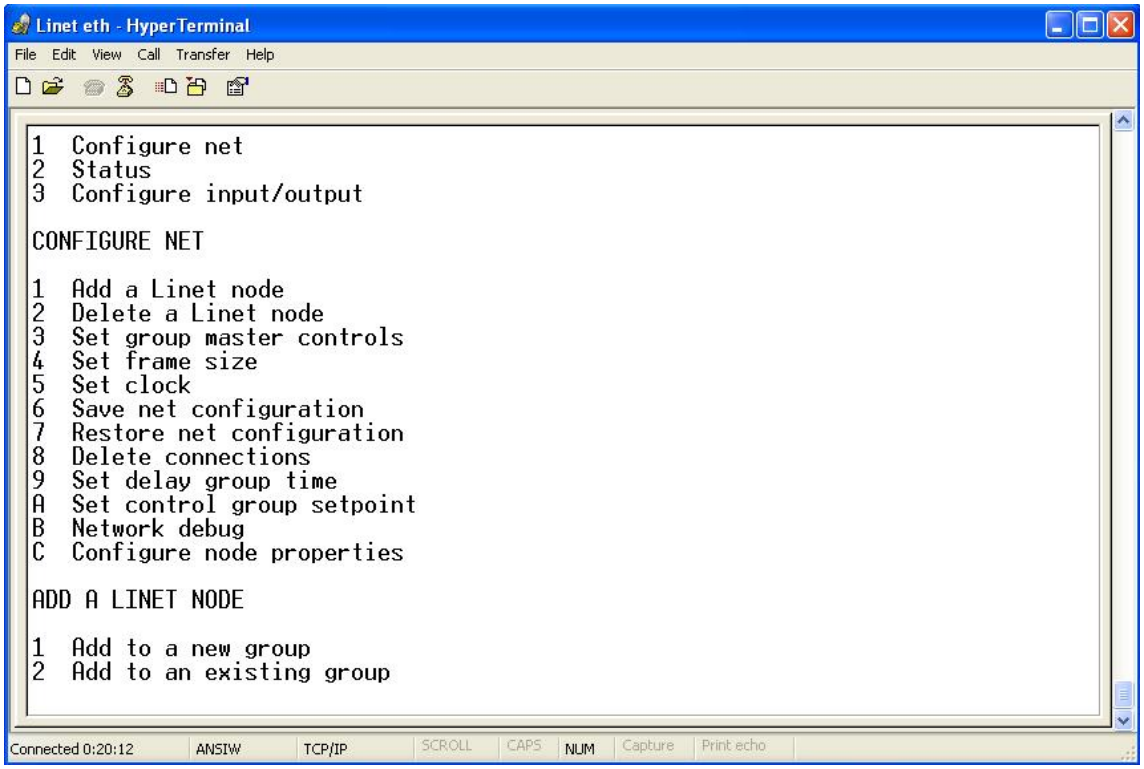


Figure A.14: Add a linet node

...then '1' (Add to a new group)...

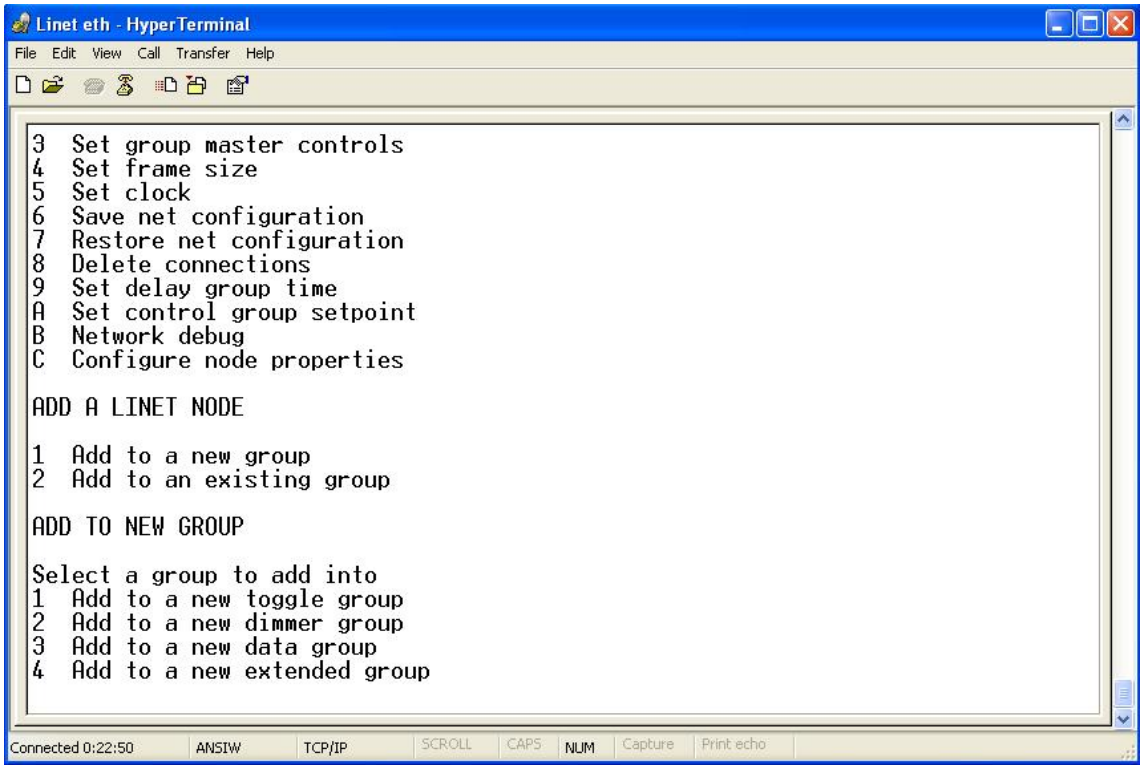


Figure A.15: Add to new group

...then '3' (Add to a new data group)....

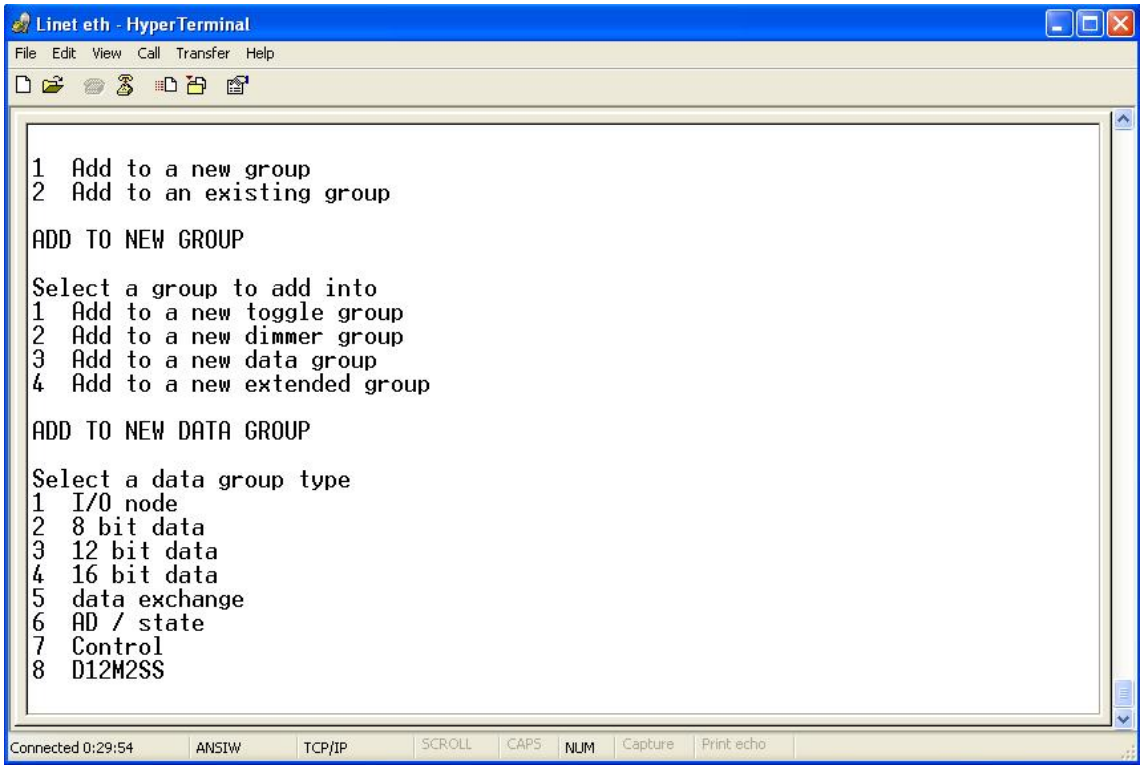


Figure A.16: Push buttons of nodes to be added to this group

...then '1' (I/O node)....

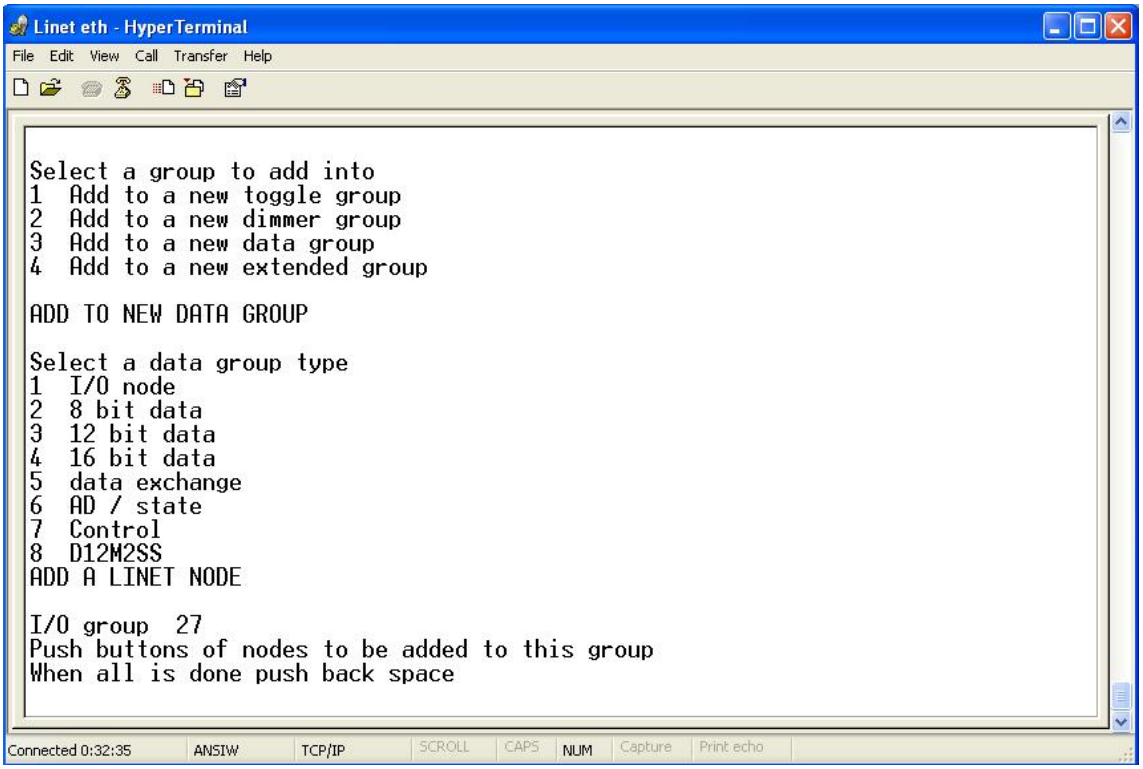


Figure A.17: Push buttons of nodes to be added to this group

Now the controller asks the user to indicate which nodes to add to this group. We then press the pushbutton switch.

While pressing the button, the indicator led on the button (when present) illuminates to indicate that the configuration has been successful. At the same time, the interface prompts 'Node 1 added'. While pressing down the button connected to the node, the system notified that the node had no network identifier, so it assigned the lowest unused identifier (nid) to it, which was '1' in this case, as the system was blank. The group has a group identifier (gid), which is also '1'.

We will do the same thing with the AD State node. This time, instead to press 1 (I/O node) in the last window, we have to press 6 (AD State) and then press the reset button of the Linet node.

A.4.2 open Facility Management Server – oFMS.

A.4.2.1 Introduction

Open Facility Management Server offers oBIX services over HTTP. It is developed to enable legacy equipments to join the network of equipments. It has been build according to oBIX specification and it implements many of the functionalities defined.

The oFMS is an HTTP server for storing oBIX information. HTTP clients communicating with the server can be user agents or equipment adapters. All the clients are able to do the same things by using the services provided by

oFMS. They can add their own data to the oFMS, they can read data, write data, or invoke operations. All the main services available can be read from the *Lobby* object (in oBIX everything is an *object*) in a server. *Lobby* object address is the only thing that client has to know when connecting to the server. It tells all the other important addresses.

All oBIX servers must provide an object which implements obix:Lobby. The *Lobby* object serves as the central entry point into an oBIX server, and lists the URIs for other well-known objects defined by the oBIX specification.

For further information about oFMS see chapter five of the manual (Introduction to oFMS Features).

A.4.2.2 Configuring oFMS

To manage the oFMS, there are two additional programs, Setup and Reset. Setup program can be used to configure server parameters, for example HTTP server port number. Setup program should be used when oFMS is not running. Reset program is used to reset the database. It removes the old database file and initializes it again using the parameters configured with the Setup program. Both Setup and Reset are command line programs. The figure A.18 shows which parameters are configurable.

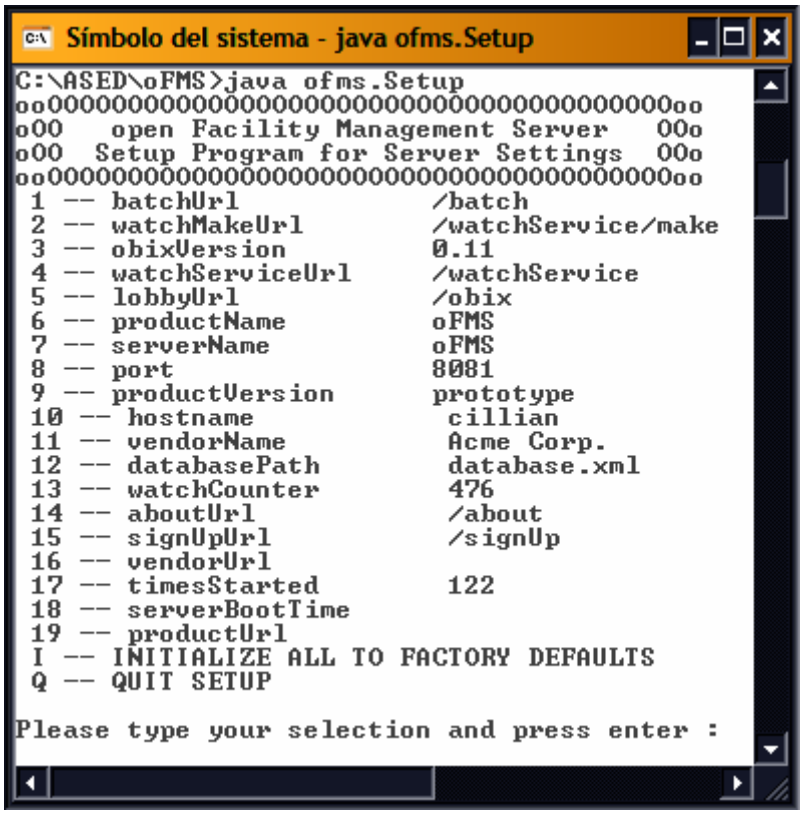


Figure A.18: Setup menu

A.4.2.3 Running oFMS

To start oFMS, simply open the file `oFMS.bat` from the directory `C:\ASED\startUp\` and you should see a screen like the one shown in Figure A.19.

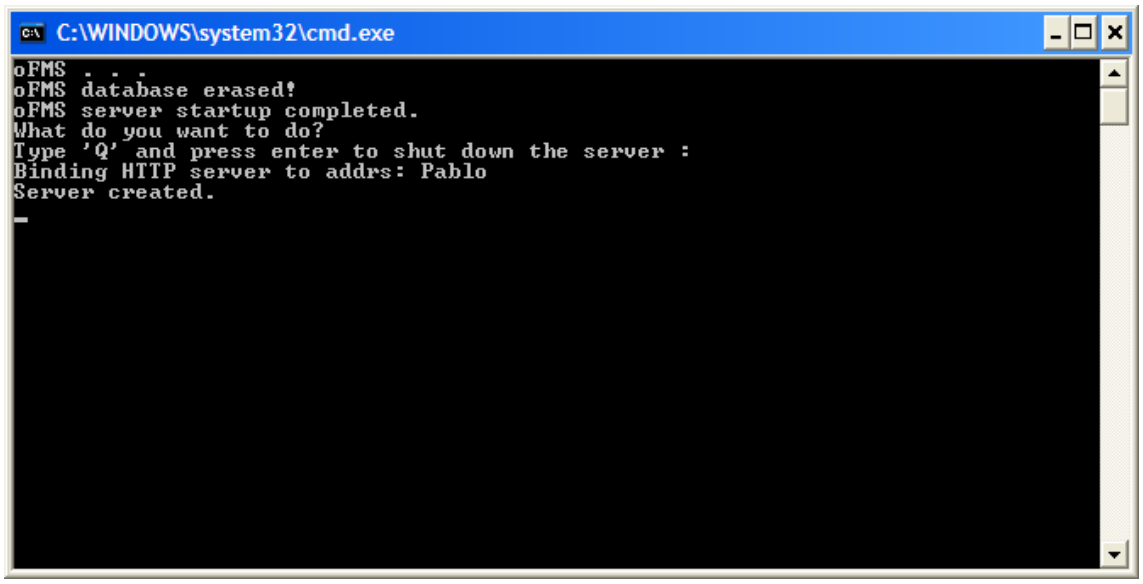


Figure A.19: oFMS screen

By default, oFMS starts on port 8081. Point your browser to http://<hostname or IP address>:8081/ and you should see a screen like the one shown in Figure A.20.

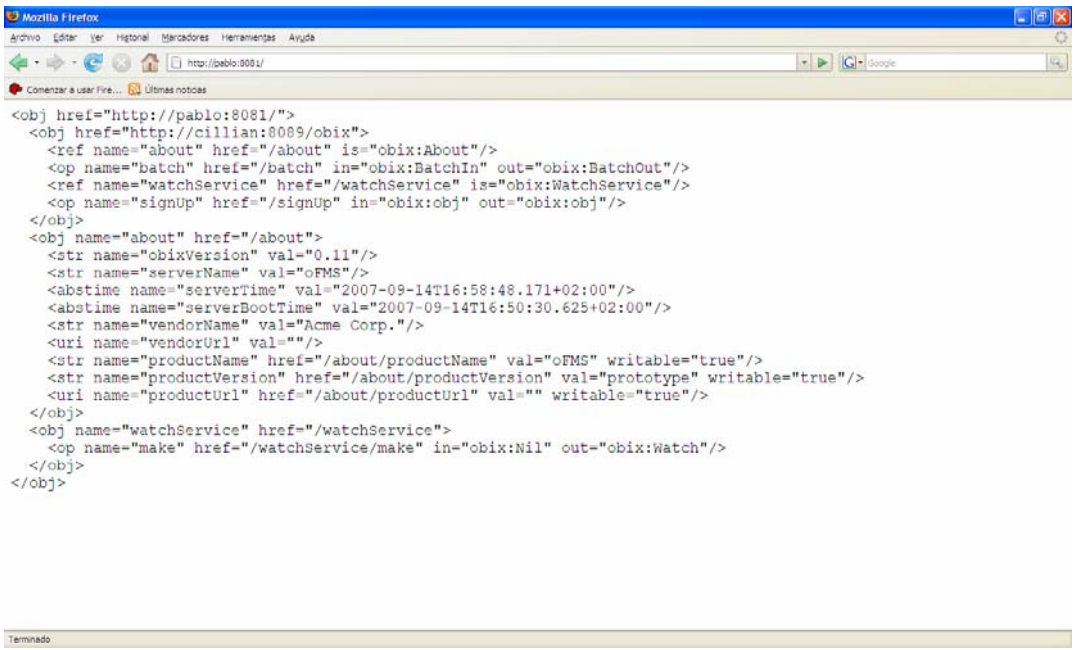


Figure A.20: oFMS server

A.4.3 Linet2oBIX Adapter

A.4.3.1 Configuring the Linet2oBIX Adapter

The Linet2oBIX Adapter is configured via and XML document as can be seen in the figure A.21. By default, Linet2oBIX Adapter looks for the `settings_<project_name>.xml` file in the `C:\ASED\Properties\` directory.

Figure A.21: *settings_example.xml* file

However, a specific configuration file can be specified in the file `Linet2oBIX Adapter.bat` from the directory `C:\ASED\startUp\` as shown in Figure A.22. Moreover, you have to write in this file the proper name of the file `settings_<project_name>.xml` when you change the name.

Usage:

<<java adapter.main_control.LinetAdapter
-p:C:/ASED/Properties/settings_lab.xml
-s:C:/ASED/Properties/properties_address.xml>>[2]

[2] `properties_address.xml` file is used to transmit `settings_<project_name>.xml` location to eCaregiver WS.

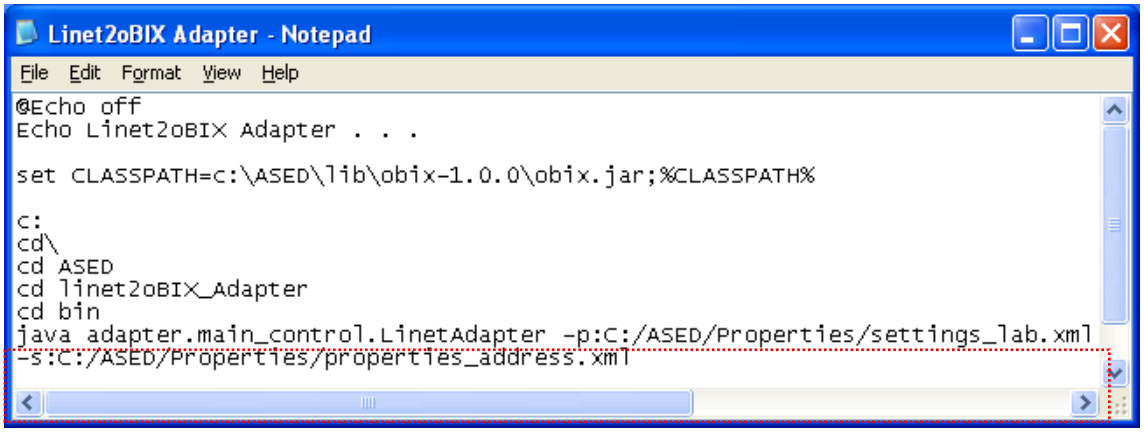


Figure A.22: *Linet2oBIX Adapter.bat* file

The `settings_<project_name>.xml` file is used to configure Linet2oBIX Adapter and whole system at the same time. A sample config file, called `settings_example.xml` (see Figure. A.21) is provided in the `\Properties` directory of this distribution.

As example, we will use the `settings_example.xml` to explain how to change this file. The top level element is "properties" which contains subelements used to configure the application. The subelements of the file are:

- "moduleName": The name of the project.

- "linetHostAddress": The host address of Linet network. For example, 130.233.120.94.

- "linetPort": The port of Linet network. For example, 1313.

- "serverAddress": The URL of the oFMS server would be http://<hostname or IP address>:8081/.

- "hostName": The host name of the computer which ASED is running.

- "hostAddress": The host address of the computer which ASED is running.

- "port": The oFMS port. By default, oFMS starts on port 8081

- "databasePath": The path of oFMS database. By default is `ofms/database.xml`.

- “numberOfGroups": The number of the groups in the Linet network.

- "displayName_group#": This is the human readable name of the group #.

- "isOnOffOutput_group#": This sub-element would be “yes” when the correspondence group would be an output.

- "timesStarted": The time started of the system. This will be the prefix of the system when it is downloaded

- "serverBootTime": The server boot time.

A.4.3.2 Running the Linet2oBIX Adapter

To start Linet2oBIX Adapter, simply open the file `Linet2oBIX Adapter.bat` from the directory `C:\ASED\startUp\` (Previously, you have to run oFMS). Point your browser to http://<hostname or IP address>:8081/ and you should see a screen like the one shown in Figure A.23 with information related with all the nodes connected to the Linet network.

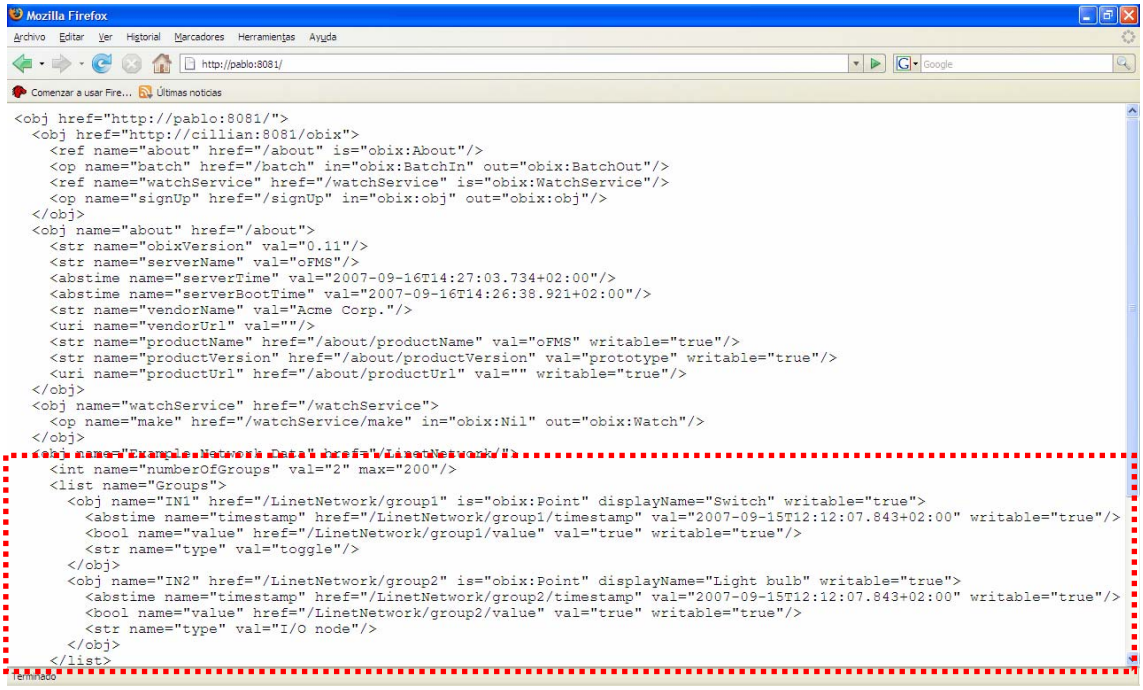


Figure A.23: Linet2oBIX Adapter screen

A.4.4 Network Simulator

A.4.4.1 Configuring the Linet2oBIX Adapter

The Linet2oBIX Adapter is also configured via and XML document. The configuration of this properties files are explained in the previous section A.4.3.1. By default, Network Simulator looks for the `settings_lab_sim.xml` file in the `C:\ASED\Properties\` directory.

Here it is also necessary to write the TYPE of each node, in this implementation is is possible to install only “I/O node” and “AD/state” nodes.

Examples:

<entry key="type_group3">AD/state</entry>

<entry key="type_group4">I/O node</entry>

A.4.4.2 Running the Network Simulator

To start Network Simulator, open the file Network Simulator.bat from the directory C:\ASED\startUp\ (Previously, you have to run oFMS). If everything is correctly installed, two windows appear on your desktop as in the Figure A.24. Point your browser to http://<hostname or IP address>:8081/ and you should see a screen like the one shown in Figure A.23 with information related with all the nodes connected to the Linet network.

Figure A.24: Network Simulator windows

A.4.5 Activities Recognition Web Service

A.4.5.1 Introduction

Activities Recognition WS is the core of the whole system. It is an HTTP client to the oFMS and handles sensor information so as to provide real time occupancy map on computer. Activities Recognition WS collects data from

oFMS via watches, process the algorithms for detecting predefined activities and log the collect data into a MySQL database.

A.4.5.2 Running Activities Recognition Web Service

To start Activities Recognition Web Service, simply open the file ActivitiesRecognitionWS.bat from the directory C:\ASED\startUp\ . Previously, if you are working with the real network you have to open the next files in the following order,

- `oFMS.bat` in the directory `C:\ASED\startUp\`

- `Linet2oBIX Adapter.bat` in the same directory,

Otherwise if you want to work with a virtual network, run the Network simulator instead of the Linet2oBIX Adapter:

- `Network Simulator.bat` in the directory `C:\ASED\startUp\`

After executing the ActivitiesRecognitionWS.bat file you should see a screen like the one shown in Figure A.25. In this window you can activate the Restart Database option. If you do not activate this option the new entries stored in the database will be stored after the entries wich were already. In case this option is activated the previous entries will be deleted.

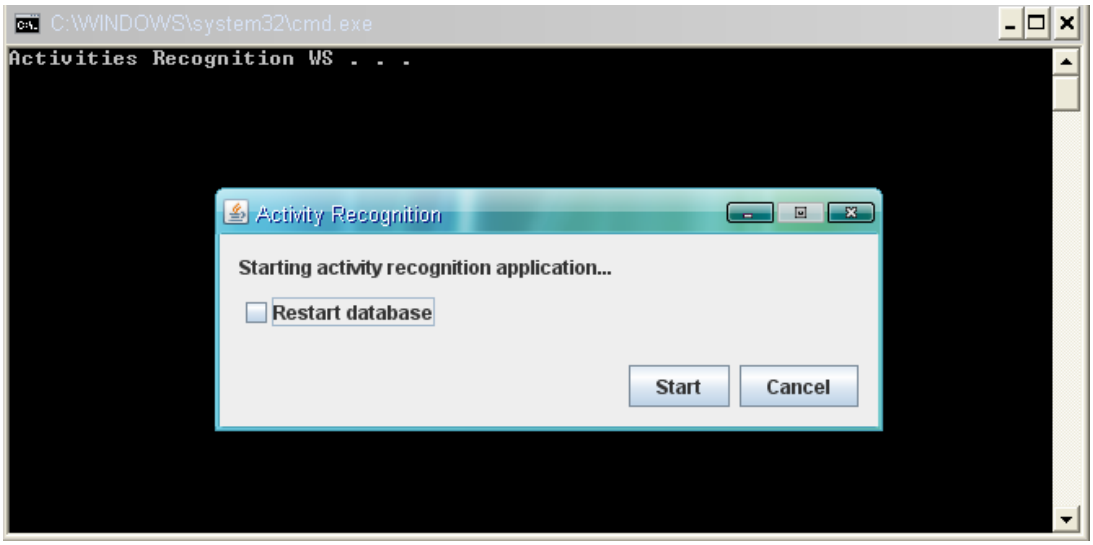


Figure A.25: Restart database option

After pressing start you should see a screen like the one shown in Figure A.26.

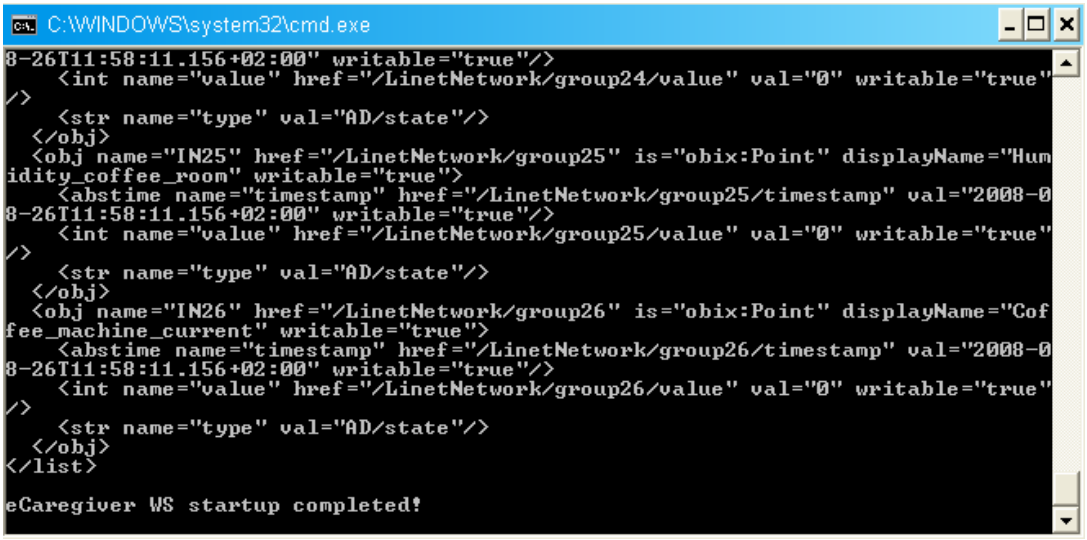


Figure A.26: Activities Recognition WS screen

A.4.5.3 Configuring the Activities Recognition Web Service

A.4.5.3.1 Adding a new device

First the device and its characteristics should be included into the properties xml file of the project.

Second the devicehas to be supported into the preprocessing module (even if it does not need preprocessing) because the application store the event in the database from the preprocessing module and also the activity analysis is executed from this module.

A.4.5.3.2 Adding a new activity

In order to adding a new activity first the sequence of events that determine this activity has to be found.

When the sequence is fixed the activity analysis algorithm has to be completed with this new activity.

If the algorithm needs to send queries to the database probably you will need to write some new specific functions in the Database class. Remember that this class is the only one that connects with the MySQL database.

A.4.6 Algorithms Tester

A.4.6.1 Running Algorithms Tester

To start the “Algorithms Tester” , open the file Algorithms Tester.bat from the directory C:\ASED\startUp\. The nodes_log table of the MySQL database has to

be created before running the Algorithms Tester (by running the Activities Recognition WS or by importing the data from an external file).

A.4.6.2 Configuring Algorithms Tester

This module can run an algorithm. The Tester gets the events from the nodes_log table of the database instead of get them from the oFMS server. The algorithm can be mainly a copy of the Algorithm module of the Activities Recognition WS application, but some adaptations are needed:

- The sequence ac.eCWS.db has to be replaced by tester in the code of the Algorithm contained in the Algorithms Tester.
- If the Activities Recognition has any new table, it is necessary to create and initialize them also in the AlgorithmTester class.
- All the functions of the Database module of the Activities Recognition WS used by the Algorithm have to be copied to the AlgorithmTester class. In all these funcions it is necessary to replace:

```
st=conn.createStatement();
rs=st.executeQuery("SELECT * FROM "+tableName+" where ...
```

with this:

```
String currentTime = timeMillis(table_name_1, p.getTimestamp().encodeVal());
st=conn.createStatement();
rs=st.executeQuery("SELECT * FROM "+tableName+" where
      "+tableName+".Milliseconds<= "+currentTime+" AND ...
```

For example

```
st=conn.createStatement();
rs=st.executeQuery("SELECT * FROM "+tableName+" where "+tableName+".Name =
      '"+nodeName+"'");
```

with this:

```
String currentTime = timeMillis(table_name_1, p.getTimestamp().encodeVal());
st=conn.createStatement();
rs=st.executeQuery("SELECT * FROM "+tableName+" where
      "+tableName+".Milliseconds<= "+currentTime+" AND "+tableName+".Name =
      '"+nodeName+"'");
```

A.5 Introduction to oFMS Features

The following section is to a large extent compiled from direct quotations of the "Introduction to oFMS Features" by Hannu Järvinen. [Jä07]

A.5.1 Introduction

Open Facility Management Server offers oBIX services over HTTP. It is developed to enable legacy equipments to join the network of equipments. It has been build according to oBIX specification and it implements many of the functionalities defined.

The idea of oFMS is to offer an oBIX interface to equipments, user agents and other oBIX servers. Server is basically just storing information in oBIX format and offering an interface for accessing it.

The oFMS is an HTTP server for storing oBIX information. HTTP clients communicating with the server can be user agents or equipment adapters. All the clients are able to do the same things by using the services provided by oFMS. They can add their own data to the oFMS, they can read data, write data, or invoke operations. All the main services available can be read from the *Lobby* object (in oBIX everything is an *object*) in a server. *Lobby* object address is the only thing that client has to know when connecting to the server. It tells all the other important addresses.

All oBIX servers must provide an object which implements obix:Lobby. The *Lobby* object serves as the central entry point into an oBIX server, and lists the URIs for other well-known objects defined by the oBIX specification.

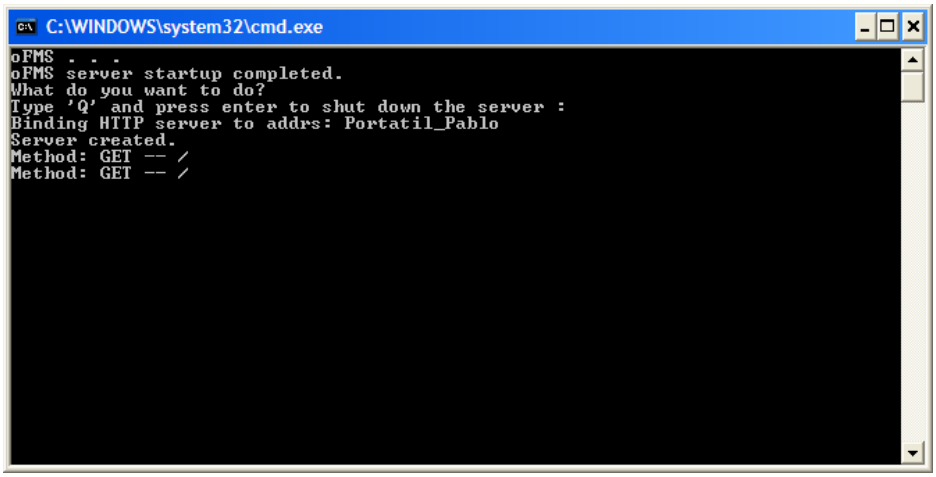


Figure A.27: oFMS text display

A.5.2 HTTP Methods

Server is using HTTP -protocol for network communication. HTTP defines GET, PUT and POST methods. In oBIX these three methods are mapped to Read, Write and Invoke Requests.

A.5.2.1 Read

Read request is used to read any object which has an address (URI). This means an object that has an href.

A.5.2.2 Write

Write request is used to change any object that has an href and writable tag is true. Equipment can thus modify its own data on the server to update it according to the current state. User Agent also uses write request to change the state of an equipment.

A.5.2.3 Invoke

Invoke request is used to invoke any op object. New equipment makes a new watch with an invoke request and adds its own data to it. That is how the equipment knows if it should change its state. This happens when some other client, for example User Agent, has made changes to the equipment data in the oFMS by using write request. To see if changes have been made, clients poll the watch URI using pollChanges operation.

A.5.3 Lobby

Only thing that client has to know when connecting to the server is the address of the Lobby object. By convention the address is http://server/obix, but servers are also free to use any other address. By reading the Lobby object, clients can determine other important addresses. In addition to features defined in oBIX specification oFMS Lobby object defines SignUp address. This address is used by equipments when registering to the server for the first time. Actual operation is adding the data about the equipment to the database.

A.5.4.1 Example

Equipment adapter is started and it reads the Lobby object to find out addresses of the important objects and services.

```
Command:
client.read("http://server/obix")

Response:
1 <obj name="lobby" href="http://server/obix">
2     <ref name="about" href="/about" is="obix:About"/>
3     <op name="batch" href="/batch" in="obix:BatchIn" out="obix:BatchOut"/>
4     <ref name="watchService" href="/watchService" is="obix:WatchService"/>
5     <op name="signUp" href="/signUp" in="obix:obj" out="obix:obj"/>
6 </obj>
```

After this it can directly use basic functionalities by using these addresses: About, BatchIn, WatchService and signUp.

A.5.5 About

About object contains general information about the oBIX server.

A.5.5.1 Example

In some cases there can be need to know more about the oBIX server itself. For example in more advanced equipment adapters there can be support for different oBIX versions. Currently supported version number can be found in servers about object. In this example equipment adapter reads the Lobby object for server related information.

```
Command:
client.read("http://server/about")

Response:
1 <obj name="about" href="http://server/about">
2       <str name="obixVersion" val="0.11"/>
3       <str name="serverName" val="oFMS"/>
4       <abstime name="serverTime" val="2006-11-27T12:24:07.749+02:00"/>
5        <abstime name="serverBootTime" val="2006-11
        27T12:24:03.031+02:00"/>
6       <str name="vendorName" val="Acme Corp."/>
7       <uri name="vendorUrl" val=""/>
8       <str name="productName" href="/about/productName" val="oFMS"
        writable="true"/>
9       <str name="productVersion" href="/about/productVersion" val="beta"
        writable="true"/>
10      <uri name="productUrl" href="/about/productUrl" val="" writable="true"/>
11 </obj>
```

At line 4 equipment adapter can see the current time on the server. At line 5 is the time when the server was last booted.

A.5.6 Batch

Batch operation can be used to bunch network requests together. All the requests are packed into batch object which is sent to the server. Response object contains responses for all requests.

A.5.6.1 Example

Lets consider an example about equipment which acts like an alarm clock. At a given time equipment wants to wake up user by turning on the television. When turning on the TV it also makes sure that it is not muted. The equipment adapter could easily make two requests to the server and do the same thing but just to minimize the network load it has been implemented to use batch operation.

```
Command:
client.invoke("http://server/batch",
1 <list name="in" is="obix:BatchIn">
2     <uri is="obix:Read" val="/tv/muteSwitch" />
3     <uri is="obix:Write" val="/tv/onOffSwitch">
4       <bool name="in" val="true" />
5      </uri>
6 </list>);

Response:
7 <list is="obix:BatchOut">
8     <bool name="muteSwitch" href="/tv/muteSwitch" val="false"
      writable="true"/>
9     <bool name="onOffSwitch" href="/tv/onOffSwitch" val="true"
10      writable="true"/>
11 </list>
```

The alarm clock equipment can now read from the response message that the TV is not muted and is now turned on.

A.5.7 WatchService

Watches offer a simple way to poll changes in certain objects. The goal is to make it simple and minimize network load. When using watches only the changes are sent in response message and unchanged objects are ignored. All the available watch operations are defined in oBIX specification. In addition all the URIs added to the watch are listed in a child object of a watch.

A key requirement of oBIX is access to real-time information. We wish to enable clients to efficiently receive access to rapidly changing data. However, we don't want to require clients to implement web servers or expose a well-known IP address. In order to address this problem, oBIX provides a model for client polled eventing called watches. The watch lifecycle is as follows:

1. The client creates a new watch object with the make operation on the server's WatchService URI. The server defines a new Watch object and provides a URI to access the new watch.

2. The client registers (and unregisters) objects to watch using operations on the Watch object.

3. The client periodically polls the Watch URI using the pollChanges operation to obtain the events which have occurred since the last poll.

4. The server frees the Watch under two conditions. The client may explicitly free the Watch using the delete operation. Or the server may automatically free the Watch because the client fails to poll after a predetermined amount of time (called the lease time).

Watches allow a client to maintain a real-time cache for the current state of one or more objects. They are also used to access an event stream from a feed object. Plus, watches serve as the standardized mechanism for managing per-client state on the server via leases.

A.5.7.1 Example

In this example we jump to the other side to think about the equipment adapter of the TV set from the previous example. The TV set should know when to turn itself on. The adapter has already added its data to the server and now wants to poll it for the changes. First the adapter has to read the watchService object.

```
Command:
client.read("http://server/watchService");

Response:
1 <obj name="watchService" href="http://server/watchService">
2     <op name="make" href="/watchService/make" in="obix:Nil"
      out="obix:Watch"/>
3 </obj>
```

Now the adapter has an address to the operation that is able to create a watch. Creating it is the next step.

```
Command:
client.invoke("http://server/watchService/make");

Response:
4 <obj name="http://server/watch26" href="http://server/watch26"
is="obix:Watch">
5     <reltime name="lease" href="http://server/watch26/lease" val="PT0S"
      writable="true"
6     min="PT60S"/>
7     <op name="add" href="http://server/watch26/add" in="obix:WatchIn"
8      out="obix:WatchOut"/>
9     <op name="remove" href="http://server/watch26/remove"
in="obix:WatchIn"
10    out="obix:obj"/>
11    <op name="pollChanges" href=http://server/watch26/pollChanges
      in="obix:obj"
12     out="obix:WatchOut"/>
13    <op name="pollRefresh" href=http://server/watch26/pollRefresh
      in="obix:obj"
14    out="obix:WatchOut"/>
15    <op name="delete" href="http://server/watch26/delete" in="obix:obj"
      out="obix:obj"/>
16    <list name="uris" href="http://server/watch26/uris"/>
17 </obj>
```

Server creates a new empty watch and returns it as an HTTP response. Now it is time to add all the addresses that adapter wants to watch. In this example the adapter watches the changes in muteSwitch and onOffSwitch objects.

```
Command:
client.invoke("http://server/watch26/add",
18 <obj name="watchIn" is="obix:WatchIn">
19 <list name="hrefs">
20 <uri val="/tv/muteSwitch" />
21 <uri val="/tv/onOffSwitch" />
22 </list>
23 </obj>);
```

The server returns current values of the objects.

```
Response:
24 <list name="values">
25 <bool name="muteSwitch" href="/tv/muteSwitch" val="false"
writable="true"/>
26 <bool name="onOffSwitch" href="/tv/onOffSwitch" val="false"
writable="true"/>
27 </list>
```

Now the adapter periodically polls changes in the objects by polling only the watches pollChanges address. As the line 28 shows no changes in these objects came up this time.

```
Command:
client.invoke("http://server/watch26/pollChanges");

Response:
28 <list name="watchOut"/>
```

A.5.8 SignUp

By signing up clients can add their own data to the server. Sign-up operation is not defined in oBIX specification but is an addition to the feature set offered by oFMS. The idea is that an equipment first writes its own data to the server by signing up and then polls changes in that data. Other clients or servers can control the equipment by altering the data on the server.

A.5.8.1 Example

In this example we have an equipment adapter of the one-room lamp control unit. After the adapter has read Lobby it wants to add its own data to the server so that it can be controlled by any other client or server. It invokes the SignUp operation.

```
Command:
client.invoke("http://server/signUp",
1 <obj name="bedroomLampDevice" href="/bedroom/lampDevice">
2   <obj href="/bedroom/lampDevice/lamp1" displayName="Desk lamp"
    is="obix:Point"
3   name="4R1" writable="true">
4     <bool href="/bedroom/lampDevice/lamp1/switch" displayName="Lamp
5       Switch" name="4R1Value" val="" writable="true"/>
6       <obj name="status">
7         <bool name="overridden" val="false"/>
8         <bool name="disabled" val="false"/>
9         <bool name="fault" val="false"/>
10        <bool name="down" val="false"/>
11        <bool name="inAlarm" val="false"/>
12        <bool name="unackedAlarm" val="false"/>
13        <bool name="historyStart" val="false"/>
14        <bool name="historyEnd" val="false"/>
15      </obj>
16    <abstime name="timestamp" val="1970-01-01T02:00:00.000+02:00"/>
```

```
17  </obj>
18 </obj>);
```

Server returns the added data object.

```
Response:
19 <obj name="bedroomLampDevice" href="/bedroom/lampDevice">
20  <obj name="4R1" href="/bedroom/lampDevice/lamp1" is="obix:Point"
21 displayName="Desk lamp" writable="true">
22    <bool name="4R1Value" href="/bedroom/lampDevice/lamp1/switch"
      val="false"
23    displayName="Lamp Switch" writable="true"/>
24      <obj name="status">
25        <bool name="overridden" val="false"/>
26        <bool name="disabled" val="false"/>
27        <bool name="fault" val="false"/>
28        <bool name="down" val="false"/>
29        <bool name="inAlarm" val="false"/>
30        <bool name="unackedAlarm" val="false"/>
31        <bool name="historyStart" val="false"/>
32        <bool name="historyEnd" val="false"/>
33      </obj>
34    <abstime name="timestamp" val="1970-01-01T02:00:00.000+02:00"/>
35  </obj>
36 </obj>
```

Now the added data on the server can be read or manipulated normally.

```
Command:
client.read("http://server/bedroom/lampDevice/lamp1");

Response:
37 <obj name="4R1" href="http://server/bedroom/lampDevice/lamp1"
is="obix:Point"
38 displayName="Desk lamp" writable="true">
39    <bool name="4R1Value" href="/bedroom/lampDevice/lamp1/switch"
      val="false"
40    displayName="Lamp Switch" writable="true"/>
41    <obj name="status">
42      <bool name="overridden" val="false"/>
43      <bool name="disabled" val="false"/>
44      <bool name="fault" val="false"/>
45      <bool name="down" val="false"/>
46      <bool name="inAlarm" val="false"/>
47      <bool name="unackedAlarm" val="false"/>
48      <bool name="historyStart" val="false"/>
49      <bool name="historyEnd" val="false"/>
50    </obj>
51 <abstime name="timestamp" val="1970-01-01T02:00:00.000+02:00"/>
52 </obj>
```

Appendix B: Linet Description

This chapter is compiled to direct quotation of the following sources: [Li08], [LP00], [LM02] and [LC03].

B.1 Network Description

The Linet [Li08] (from Light Control Network) network consists in a controller and nodes [LP00]. The nodes have entry ports where sensors and actuators are connected. Those sensors and actuators can be of any kind, as far as they provide an electrical signal that can be read by the nodes. A single controller can control up to 200 nodes. One of the advantages of the Linet network is that several nodes can be connected to the controller using a single pair of wires. The controller maintains a table of rules defining the interactions between sensors and actuators. This is another advantage since it avoids a voluminous logic wiring, especially if there are many devices and conditions. The rules can therefore be very complex and cost very few efforts. The wiring is even more reduced since the nodes can provide power supply to the sensors/actuators connected to them. The nodes are separated within groups [LM02]. Several kinds of groups are available.

The two following tables shortly define them:

Basic groups		
Group type	Description	Application example
Toggle	Each node within a toggle group has a bi nary state, which is common to all nodes within the group. The state is inverted when there is a rising edge detected on a switch input on any of the nodes within the group.	lighting group
Analog Input	Each node contains an integrated 12bit A/Dconverter with internal, 1.25V reference. When in use, the converter performs the conversion and feeds the resulting 12bit figure to the network.	temperature sensors
Dimmer	Each node within a dimmer group outputs a state, which is common to all nodes within the group. An input switch connected to any of the nodes may be used to control the output.	lighting dimmer groups
Data group	Each node is capable to receive and send binary data at constant rate. Data exchange with the network controller can be done with 8, 12 and 16 bits words. Data exchange group consists in two nodes exchanging data between themselves.	communication units

Table B.1: Basic Linet devices groups [LP00]

Additional groups		
Group type	Description	Application example
I/O-node	Toggle node is an individual I/O-device, connected to one input and one output.	lighting master switch
Lamp	Lamp nodes are used to control a load. They may be used as slave to dimmer or toggle groups.	principal lighting systems
Call	A call node puts its output to ON and generates a “call at gid” message on the controller. Once it receives the acknowledgment, it turns its output OFF.	alarm systems, hostess call systems
AD/state	The AD/state node is an analog input switch node.	combined humidity and temperature node
Control	Control groups are used to replace thermostats when controlling temperature (or other magnitude). An input value coming from other sensors, applied to the control, is converted to centigrades.	heat control systems.

Table B.2 Additional Linet devices groups [LP00]

Delay groups are also available. They allow introducing a delay between sensor reaction and actuator effect.

The Linet controller are connected through a standard Ethernet connection. This connection allows both network monitoring and network configuration. The same information is also available through a serial link, but is more difficult to implement and not easy to connect in a house (distance matters).

B.1.1 Linet Nodes

Linet node is a fully functional network adapter to be used in the Linet network system. In the LINET network you can connect up to 200 nodes in one network with up to 200 meters of twisted-pair cabling.

LINET network uses time division protocol. All nodes have their own time slot in the signal frame. Each frame starts with a synchronization field, followed by system service bits. Next comes the 200 data bits, one for each node. The total frame length is 253 bits. During one frame each node can send and receive one bit of information. Each node can normally transmit and receive the full 80 bits/s all the time. The carrier frequency of the network is 20 kHz so the system full duplex data rate is about 200x80 bits/s. The nodes have always a constant system delay and constant data throughput rate.

You can change the network frame size and system speed. The default full duplex data rate is 80 bits/s for each node simultaneously, and the maximum number of nodes in the network is 200 pcs. If higher data rate is required, the rate can be increased to 160 bits/s or 320 bits/s by lowering the frame size of the network to 100 or 50 nodes, respectively.

B.1.2 Linet Controller

Linet network controller is a stand-alone network power supply and controller to be used in the LINET 'Light Control Network' system. “It contains a LINET bus driver, an onboard RISC microcontroller and firmware for running the standard network functions, an RS232/RS485 serial interface, an Ethernet interface for connection to industrial Ethernet or intranet systems, and an extension bus for a local microcomputer”. [LC03]

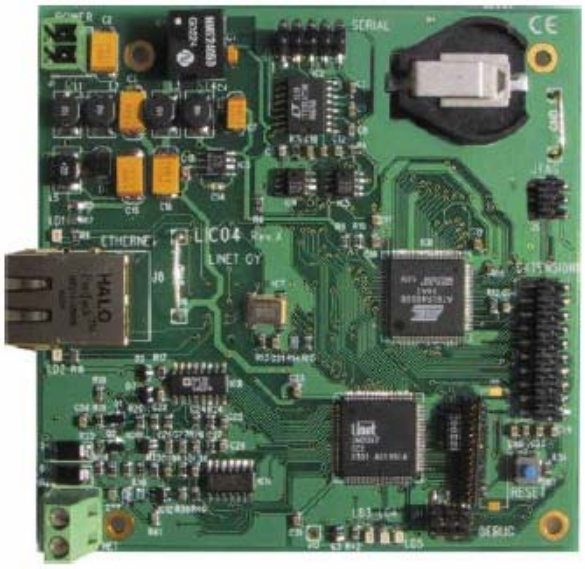


Figure B.1: LIC04 Linet Network Controller

B.1.2.1 General Information

The LIC04 controller software tasks are:

- to control the groups within the Linet system.

- to control the host interfaces and their protocols.

- to offer services for setting up the network.

B.1.2.2 Host Interface: Ethernet

- Protocol overview

The UDP/IP protocol is run on the 10BASE-T Ethernet connection to the controller.

- Controller addressing

The controller uses one UDP port. The port number and IP address of the controller are configured during controller setup.

- Request/response cycle

The controller sends the sensor-information only on request from the Linet network. A response is sent for each request packet. The response packet is sent to the IP address and port marked as sender in the request packet.

The controller can respond to multiple requesters. The requests are processed in sequence; a new request is processed only after the previous request is responded to. If the rate of request exceeds the possible response rate, the extra request is silently discarded.

- Data types

The controller receives and sends packets in binary format. Data entities are 8 or 16 bits unsigned integers. The same data structure is used to send and receive data to and from the controller, only one bit changes to indicate if the packet is a request or an answer, or to indicate of we want to update a node or read it.

A transmission consists of a header that gives global information and 200 network status packets describing each node of the network.

B.2 Protocol

The controller accepts UDP/IP connections and only transmits data on demand [LS02]. The controller receives and sends packets in binary format. Data entities are 8 or 16 bits unsigned integers. The same data structure is used to send and receive data to and from the controller, only one bit changes to indicate if the packet is a request or an answer, or to indicate if we want to update a node or read it. A transmission consists of a header that gives global information and in 200 network status packets describing each node of the network. Transmissions with less network status packets are possible to increase the available response rate of the controller, but, as in our case, only one computer is connected to the controller, it has little interest.

B.2.1 Packet Description

Every transmitted packet starts with the following common header:

byte offset	bytes	field
0	1	protocol version
1	1	packet type, see below
2	2	flags, reserved

Table B.3: Linet packet header [LS02]

The defined packet types (second byte) are defined in Table 4.4:

0	status request
1	status response
2	structure request
3	structure response

Table B.4: Linet packet types [LS02]

The provided protocol description doesn't contain any further information about the structure request and response.

byte offset	bytes	Field
0	1	group type, see below
1	1	flags
2	2	group value

Table B.5 Network status packet [LS02]

The flags field contains only one flag: if bit 0 is on, the request wants to change the group state to the value contained in the packet, if the flag bit is off, the value field is ignored.

The group types are the ones defined earlier:

type code	group type	value type
0	NONE	none
1	TOGGLE	on/off
2	DIMMER	0=off, 1=on, 2=step, 3=step down
3	IOGROUP	1 bit I/O
4	XLAMP	1 bit I/O
5	LMON	1 bit I/O
6	XDELAY	16 bit delay value in milli sec
7	WCALL	waiter call
8	DATA EXCHANGE	not usable
9	DATA8	8 bit data value
10	DATA12	12 bit data value
11	DATA16	16 bit data value
12	AD/STATE	AD value (12 bit)
13	CONTROL	AD value (12 bit)

Table B.6: Linet Group types [LS02]

Appendix C: oBIX description

This chapter has been compiled from direct quotations of the following sources: [Ga05], [OA08], [Jä07], [Ha06] and [Co06]

C.1 History

The development of oBIX began in April 2003 as the CABA XML/Web Services Guideline Committee. The purpose was to develop publicly available web services interface specification that can be used to obtain data in a simple and secure manner from HVAC, access control, utilities, and other building automation systems, and to provide data exchange between facility systems and enterprise applications. [Ga05]

Originally project was launched at Buil Conn 2003. After that it identified a need to become an open standard and quite soon, in 2004, the development work was moved from CABA to OASIS. It made strong technical progress and had a successful public demonstration with four vendors in March 2005. [Ga05]

Version 1.0 of the specification was released in 2006. There is also publicly available JAVA open source toolkit for creating oBIX enabled applications. It is being developed parallel with the standard. [OA08]

C.2 oBIX Data Model

In oBIX data is described with a certain XML language, which can be simply referred as the oBIX language. This common XML language is the foundation of oBIX. It enables representing the information from different building automation systems in a standardized way. The object model of oBIX is simple, which makes it very easy to use. Still the extensibility through the contracts provides a convenient way to define new abstractions for more complicated objects or services. [OA08]

If an oBIX client understands the small fixed schema of the XML syntax, it can access the whole object tree whether it is able to understand all the information or not. No information is dropped out, it is all there. Responsibility of understanding all needed information is left to the client. [OA08]

C.3 Networking

In addition to a standard way for representing the data, we need means to transfer it through the network. Therefore an abstract request/response model is specified. Specification version 1.0 defines also both SOAP and HTTP REST bindings that implement the model and can be used for communication. This gives the manufacturer freedom to use either SOAP or HTTP. Of course this can also cause situations where a client supports only the one binding model that is not supported by the server. [Jä07]

Because oBIX is using traditional client-server architecture where clients do not have to implement web servers or expose an IP address, clients always have to poll for changes in the server. However, polling is not that elegant way to achieve the requirement of the near-real-time information access. Still this is implemented well in oBIX and a model for client polled eventing called Watches is provided. It decreases the amount of the unnecessary network requests and eliminates the resending of the unchanged data. [Jä07]

C.4 Architecture

Therefore oBIX is much more than just a way to describe points, historical trends, and alarms. It is an extensible model that describes other models (a meta-model). oBIX allows control vendors to fully describe their proprietary systems and allow enterprise to discover non-standard data and invent new applications for it. [Ha06]

Extendibility is woven into the very fabric of oBIX using something called the contract. A contract is a list of all the patterns a complex piece of data conforms to. Contracts are used to describe standardized structures such as points, historical trends and alarms. Its position with other oBIX schema parts is showed in Figure C.1. They are also used to describe proprietary vendor data. The beauty of the contract is that new ones can be introduced without changing the oBIX schema. [Ha06]

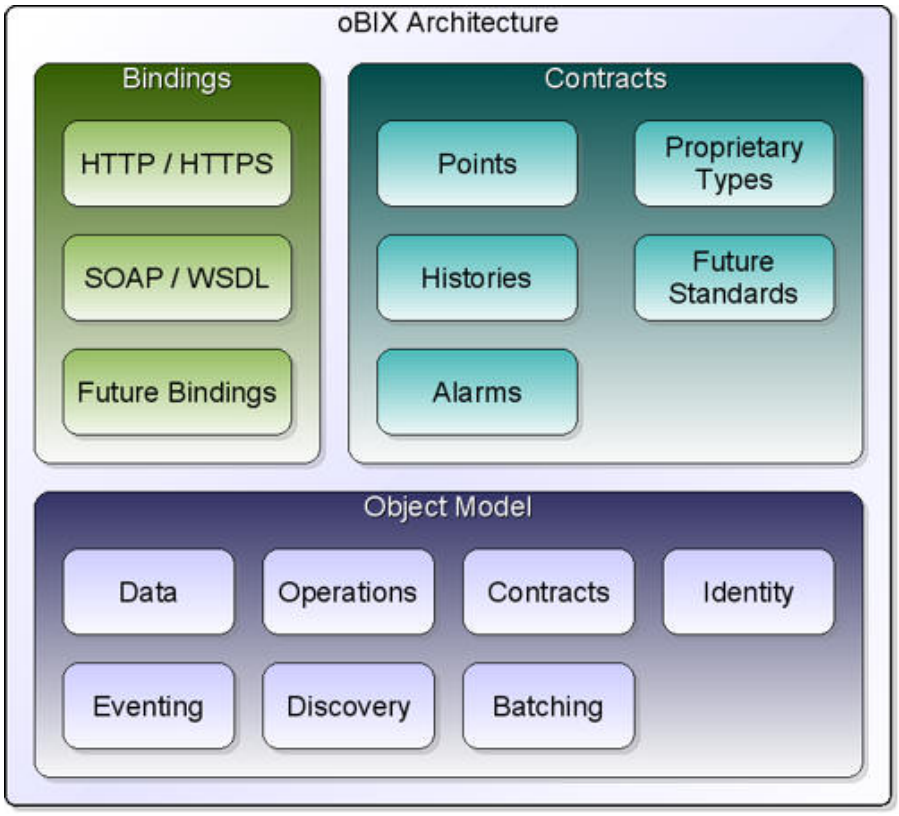


Figure C.1: oBIX architecture distinguishing bindings, contracts and objects model structure [Co06]

In typical Web services built for business applications, introducing new data structures requires new schema documents. Enterprise tools cannot handle unknown schemas so they simply ignore unanticipated data. With oBIX, vendors and standards bodies can define and even if a client doesn't know what to do with the new contracts, it can still access and work with the primitive values within. All data is first class data and will not be ignored by tools. [Ha06]

oBIX is unique in another regard, it is binding agnostic. Not only is there a SOAP binding so oBIX can interoperate with WS (the Web services stack), there is an HTTP binding making oBIX a RESTful standard. REST (REpresentational State Transfer) is the architectural style of the World Wide Web. While not a standard itself, it is best described as the set of standards that make the web successful. HTTP, URL, XML and HTML are some of those standards. oBIX is built upon these standards. It identifies objects with URLs, represents object state with XML, and transfer objects using the Hyper Text Protocol (HTTP). oBIX servers can be accessed with a web browser and therefore can be indexed by search engines, linked to by other web pages and basically interoperate with any other mainstream web technology. [Ha06]

To summarize, the oBIX architecture is based on the following principles: [Ha06]

- Object Model: a concise object model used to define all oBIX information.
- XML Syntax: a simple XML syntax for expressing the object model.
- URIs: URIs are used to identify information within the object model.
- REST: a small set of verbs is used to access objects via their URIs and transfer their state via XML.
- Contracts: a template model for expressing new oBIX "types".
- Extendibility: providing for consistent extendibility using only these concepts.

C.4.1 Data Model

All information in oBIX is represented using a small, fixed set of primitives. The base abstraction for these primitives is cleverly called object. An object can be assigned a URI and all objects can contain other objects. There are eight special kinds of value objects used to store apiece of simple information:

- *Bool*, stores boolean value, true or false.
- *Int*, stores an integer value.
- *Real*, stores a floating point value.
- *Str*, stores an enumerated value within a fixed range.

- *Abstime*, stores an absolute time value (timestamp).

- *Realtime*, stores a relative time value (duration or time span).

- *Uri*, stores a Universal Resource Identifier.

Any value object can also contain sub-objects. There are also a couple of other special types: *list, op, feed, ref,* and *err*.

These object types map one to one an XML element type. The oBIX object model is summarized in the following illustration, figure C.2. Each box represents a specific object type (and XML element name). Each object type also lists its supported attributes.

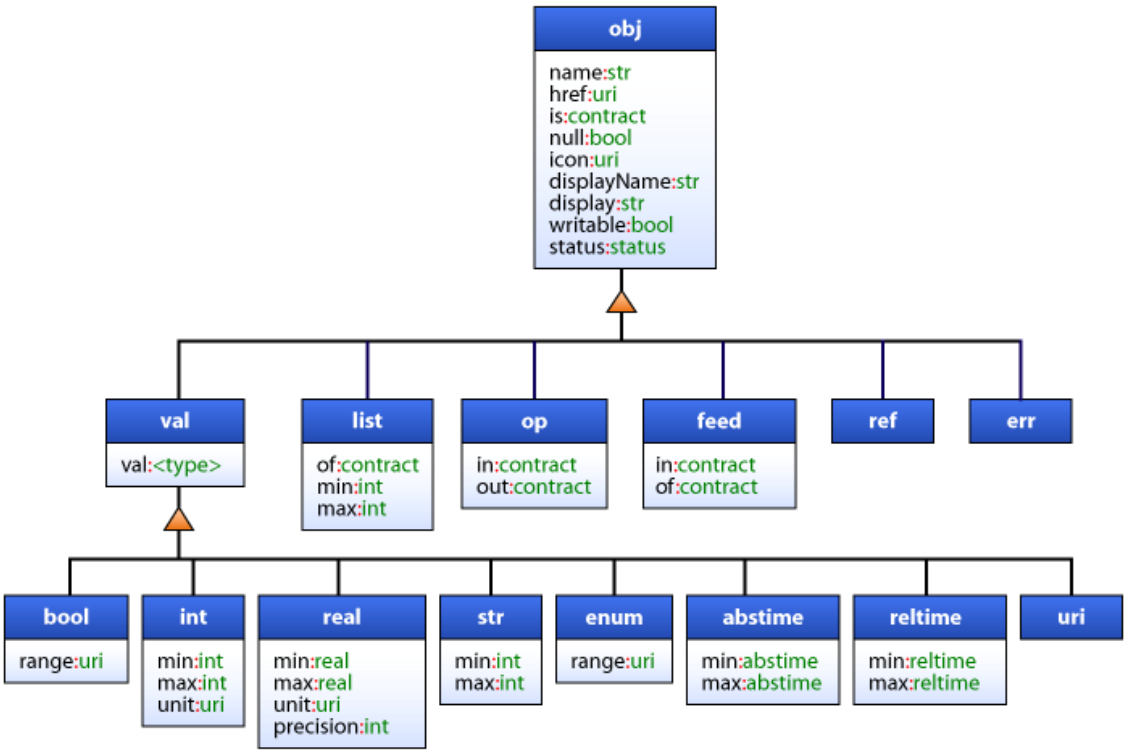


Figure C.2: Object model schema [Co06]

oBIX provides a flexible object model to describe the data and operations available on the server. In oBIX everything is an object: objects are also used to describe data types (classes) and operations (method signatures). The flexibility of oBIX is based on the possibility to custom define any kind of object. Both subtyping (is-a) and composition (has-a) are supported. [OA08]

As I already explained, oBIX follows what is called a RESTful approach (“Representational State Transfer”), a resource centric architectural style for Web Services. Central concepts of the RESTful approach are resources that share a uniform interface and a highly restricted set of operations on these resources. This approach mimics the World Wide Web where only the commands GET, PUT and POST are used to access countless resources. [OA08]

The same is true for oBIX services: only three network request types are defined at the WS level. The first two of them are read (applicable to any object) and write (for writable objects). For operations beyond basic “get” and “set,” custom operations can be defined on the oBIX level, just as any other custom object. For invoking these operations a third base network request type invokes is provided. In addition to SOAP, its RESTful approach also allows oBIX to easily support a plain HTTP protocol binding, needing nothing more than HTTP GET, PUT and POST. [OA08]

Naming in oBIX is realized via two different, complementary concepts. First, every object can have a “name.” It is used to identify a sub object within a composite object (for example, consider an object with two string members or two operations – their name tells them from each other). Second, every object can be assigned a URI (an “href”). This is necessary whenever an object is to be referenced from the outside. To apply any network request (read, write, or invoke) to an object, this reference is required. No higher-level semantics are associated with the URI namespace. [OA08]

C.4.2 Predefined Objects

The root object specifies a number of mandatory attributes like an object’s name, which are inherited by all other objects. Like in every data model, various object types to hold primitive values are defined. These are booleans, integer and floating point numbers, enumerations, strings, points in time and time spans. oBIX base object types also cover a number of universally applicable concepts: lists and feeds (containers with either static content or event queue semantics), errors, references to other objects, and operations. Custom classes can be derived from any object type. Multiple inheritance is supported. Every custom-built class can be handled by any oBIX client. oBIX provides a very flexible SI-based system for describing engineering units that is also based on objects. Furthermore, the oBIX standard library defines special purpose classes encapsulating server functionality. [Ne06]

The Lobby object has a well-known location on the server and serves as its entry point. Besides providing information about the server through the About object, it offers services to reduce protocol overhead. First, it provides a special batch operation which accepts an entire batch of requests at once. Second, clients can register objects with the watch service. Every time the client later polls this watch, the server will return a feed of events (value changes) which have happened since the last poll operation. This also ensures that no value changes are lost, independent of the polling cycle. [Ne06]

Points are classes which are used to flag primitive values as coming from the automation system. Read only points are effectively an empty class (acting merely as a semantic marker); writable points additionally specify an operation for altering the corresponding value. [Ne06]

Historical trends can be represented via the oBIX History Record, which groups a point value and a time stamp. The History object consists of a list of history records and methods to query them. Query filters can be specified and extended to a rollup calculation for, e.g., average values. [Ne06]

Eventually, oBIX defines a normalized model to query, watch and acknowledge alarms. An oBIX server supplies feeds of alarm objects. Every time the server detects that the value of an object meets a predefined alarm condition, an alarm object is added to the feed. This object contains a timestamp and points to its source. Alarms can be stateful (i.e., the point in time when the source returned to normal is recorded) and they can record if (and by whom) the alarm was acknowledged. [Ne06]

C.4.3 XML Representation

Base object types are directly mapped to individual XML elements. For example, if an object is an integer or is derived from an integer, it is rendered as <int/>. Any other objects, which are not derived from a base object type, are all rendered using the standard <obj/> element. In this case, the “is”-attribute specifies the class of an object. [Ne06]

A similar distinction exists regarding how the attributes of an object are rendered. Base attributes, e.g., the name, are mapped to XML attributes – called “facets.” Any individually added attributes are represented as sub objects nested within the opening and closing tags of their parent object. [Ne06]

Methods are, on the one hand, the normal network request types on the WS level – any object can be read, written, or in case of an operation, invoked. On the other hand, operations added on the oBIX level are again represented as sub objects. [Ne06]

Sub objects can be included in their full XML representation or via a reference. Whether composition by containment or reference is to be used has to be determined when designing the class hierarchy. For example, an oBIX alarm object is encoded as: [Ne06]

```
    <obj name="somealarm" is="obix:Alarm">
<ref name="source" href="/myhouse/somewhere"/>
  <abstime name="timestamp" val="2006-10-12_12:11:02"/>
</obj>
```

The object referenced by the “is” attribute is called a “contract.” It acts like a template for the referencing object. All sub objects of this reference object are inherited by default. Still, the referencing object can override these values (subobjects). So this object referenced by the “is” attribute defines a class (in the sense of object orientation). [Ne06]

Objects can fulfill multiple contracts (resulting in multiple inheritance). Finally, contracts can also be empty and merely describe semantics of an object – a prominent example being the (read-only) oBIX Point. In many aspects, contracts are similar to Java Interfaces. [Ne06]

An example contract describing a generic model of a furnace (a furnace template or “class”) [Ne06]:

```
<obj href="def:furnace">
    <bool name="burnerOn"/>
    <real name="curTemp" is="obix:Point"/>
    <real name="setTemp" val="50.0" is="obix:WriteablePoint"/>
</obj>
```

Note the use of point contracts (the WriteablePoint contract adds a write operation) and that a default value is specified for the second. An object following this contract: [Ne06]

```
<obj name="furnace" href="myhouse/heating/furnace" is="def:furnace">
    <bool name="burnerOn" val="true"/>
    <real name="curTemp" val="45.3"/>
</obj>
```

This would be an instance of the furnace class (an actual furnace). It inherits the default value of 50 for the set temperature. By specifying the curTemp sub object itself, any possibly inherited default value is overridden. However, the point contract for this attribute is still inherited. To allow access by an oBIX client, an href is specified. [Ne06]

Bibliography

[DM08] Definition of “Data Mining” at Wikipedia, http://en.wikipedia.org/wiki/Data_mining. (Sept. 21, 2008).

[Ja08] JavaServer Pages Technology, http://java.sun.com/products/jsp/ (Sept. 21, 2008)

[Jä07] Hannu Järvinen. “An Interoperable Equipment Server for Building Automation Systems”. Master’s Thesis, Helsinki University of Technology, 2007.

[JD08] JDBC - Java Database Connectivity Tutorials: 1. JDBC MySQL Tutorial > Mapping MySQL Data Types in Java, http://www.roseindia.net/jdbc/jdbc-mysql/mapping-mysql-data-types-in-java.shtml. (Sept. 21, 2008).

[Ra07] Pablo Ramos. “Interoperable Web Services for Home Automation for the Elderly and Disabled”. Master’s Thesis, Helsinki University of Technology, 2007.

[Li08] Linet network company home page, http://www.linetnetwork.com/

[LP00] Linet, product description, http://www.linetnetwork.com/assets/download/gendescr.pdf (Sept. 21, 2008)

[LM02] Linet. Configuration Manual, 2002, http://www.linetnetwork.com/assets/download/linconf.pdf (Sept. 21, 2008)

[LC03] LIC04 Linet Network Controller, 2003 http://www.linet-network.com/assets/download/lic04.pdf (Sept. 21, 2008)

[LS02] Linet. Software specification V6.3, 2002, http://www.linetnetwork.com/assets/download/licsw6v3.pdf. (Sept. 21, 2008)

[Ne06] Matthias Neugschwandtner,Georg Neugschwandtner, Wolfgang Kastner Interoperable Web Services for Building Automation – Integrating KNX, 2006 https://www.auto.tuwien.ac.at/downloads/knxsci06/neugschwandtner-web_services-knxsci06-website.pdf. (Sept. 21, 2008)

[OA08] OASIS Open Building Information eXchange Technical Committee. oBIX homepage. http://www.oasis-open.org/home/index.php (Sept. 21, 2008)

[Ha06] Hansen, Aaron; “oBIX Unbound”; May 2006, http://www.automatedbuildings.com/news/may06/articles/obix/060425112552obix.htm. (Sept. 21, 2008)

[Co06] T. Considine, P. Ehrlich, and B. Frank. oBIX Specification, 2006.

[Ga05] Patrick Gannon. Oasis open building information exchange technical committee. BuilConn Europe Presentation, Nov 2005.